2

PREDICTING ORGANIZATIONAL PERFORMANCE: APPLICATION OF NEUROCOMPUTING AS AN ALTERNATIVE TO STATISTICAL REGRESSION

THESIS

Franklin W. Baugh
Captain, USAF

AFIT/GEM/LSM/89S-4

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GEM/LSM/89S-4

PREDICTING ORGANIZATIONAL PERFORMANCE: APPLICATION OF NEUROCOMPUTING AS AN ALTERNATIVE TO STATISTICAL REGRESSION

THESIS

Presented to the Faculty of the School of Systems and
Logistics of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Engineering Management

Franklin W. Baugh, B.S.M.E
Captain, USAF

September 1989

Acknowledgements

I would like to thank Major Thomas Clarkson of Headquarters Strategic Air Command and Major Jeffrey Charles of Headquarters Tactical Air Command for their cooperation in making available IG report data without which this research effort would not have been possible. Also, a very big thanks to my advisor, Lt Col James R. Holt for providing the inspiration for this research. His vision, encouragement and forbearance were indispensable to the success of this effort.

Table of Contents

List of Figures

List of Tables

Abstract

This thesis presents an application of neurocomputing as an alternative function approximation tool for predicting organizational performance.

Organizational performance assessment typically involves association of component organizational attributes with a composite ordinal rating. This is a complex generalization problem which is difficult to solve by analytical means. Conventional statistical methods are particularly unwieldy for this application because of the unknown functional domain and multi-dimensional interactions among predictor variables.

The back-propagation neural network is an alternative function approximation tool which has been shown to outperform statistical regression in applications lacking a well defined domain model. This research applies this technology to the problem of predicting organizational performance.

The content and ratings of Air Force civil engineering unit effectiveness inspection reports were used to train back-propagation neural networks to discriminate between "excellent" and "satisfactory" squadrons. These trained networks significantly outperformed logistic regression models in correctly predicting squadron performance ratings during cross-validation trials (81 percent versus 61 percent).

Neurocomputing is a new, rapidly emerging technology that has only recently been explored in applications beyond signal, speech and image processing. These results demonstrate a successful application of neurocomputing as an organizational performance assessment tool. In the organizational sciences, this approach to effectiveness measurement may prove to be a useful alternative to conventional least square estimation methods. This technology can have immediate application as a management decision making tool to help maximize composite organizational effectiveness.

PREDICTING ORGANIZATIONAL PERFORMANCE:
APPLICATION OF NEUROCOMPUTING
AS AN ALTERNATIVE TO STATISTICAL REGRESSION

I. Introduction

Overview

Concern with the effectiveness, productivity, efficiency or excellence of organizations has always been a preoccupation of managers, receiving much attention in the popular and professional literature (2:539; 12:514). Much of this interest stems from the desire to recognize high quality performance and to understand how component attributes combine to yield an effective organization.

Traditional approaches to organizational performance evaluation use efficiency ratios or productivity measurements which are well defined and relatively easy to measure. However, increasing attention is being directed toward the broader construct of organizational effectiveness which is much harder to measure than process efficiency.

In many organizations, it is not clear what constitutes optimal performance. Effectiveness criteria is ambiguous and cannot be defined simply in terms of inputs and outputs or profit and loss. Public sector organizations, in particular, lack the private sector dimension of profit as an aggregate measure of effectiveness and must therefore use alternative approaches to performance evaluation (1:87-88). Effectiveness of such organizations is necessarily determined in relative terms using subjective means of evaluation. Typically, multiple measures or judgments are used to arrive at a single aggregate measure of organizational performance (12:528).

In the case of an Air Force Unit Effectiveness Inspection, such an approach is used to derive a composite organizational rating from a collection of disjoint observations (e.g. exercise response, budget compliance, etc.). Human judgement must be used to discern from these observations patterns which correspond to a given ordinal performance rating.

A means to capture this human judgement and approximate the relationship, mathematically or otherwise, between such component observations and overall performance would be useful. This modeled relationship could be a supplement to subjective evaluations and also a helpful management tool.

Such a tool would be particularly useful to commanders of base level civil engineering squadrons. Civil engineering squadrons are complex with many diversified operations. Commanders are frequently faced with conflicting objectives and must determine which objective contributes most to the success of the organization.

A method to independently derive a composite unit rating from component organizational attributes may help commanders better predict the outcome of their decisions. Such a method could indicate which course of action would yield the desired outcome. For example, some expert evaluators consider a civil engineering squadron "effective" in terms of its overall support of the host wing mission. Such support requires productive use of design and construction resources and shop craftsmen to provide and maintain host wing facilities. Also important is training of military personnel in preparation for periodic wing mobility exercises. Both objectives contribute to overall mission support; however, host wing facility support is sometimes accomplished at the expense of mobility training and vice versa. A means to independently assess composite effectiveness would permit a commander to calibrate allocation of resources toward each objective to maximize overall mission support.

Research Objective

Civil engineering organizations are composed of many branches and shops working in concert to fulfill a broad unit mission. It is possible to measure the efficiency (and, in part, effectiveness) of each component. But, it is not as clear how they combine or work in concert to yield an "excellent", "satisfactory, "marginal" or "unsatisfactory" organization as perceived by expert evaluators during "snap-shot" inspections.

Human experts recognize the relationships and patterns among organizational components which correspond to an overall *ordinal* performance rating. Can this expert judgement be captured or replicated in such a way that the ordinal measure of effectiveness may be discerned independently by analytical means? Is it possible, using past documented inspections, to approximate the functional relationships between organizational component attributes and overall unit effectiveness? Can these relationships be used to predict the rated effectiveness of an organization from subsequently collected observations?

The objective of this thesis is to show that it is possible to capture these relationships and to aggregate individual component evaluations using generalized problem solving techniques.

Functional Mappings. This type of generalization problem is a subset of a broader class of problems involving associations or "mappings" of objects from one set with objects in another set (e.g. component attributes $\longrightarrow$ ordinal ratings). Many such problems, while easy for humans, are difficult to solve analytically.

Standard AI approaches (rule-based expert systems) and conventional mathematical techniques (such as statistical regression) are particularly unwieldy for modelling organizational effectiveness. With the complexity of organizational dynamics, it would be an enormous if not impossible task to define an adequate knowledge base for an expert system application. Approximation of functional mappings or relationships between predictors and overall ratings via mathematical regression may be possible. However, such relationships may be subtle and involve multiple interactions among predictors. Furthermore, the functional form of the underlying domain model is unknown and such an approximation would likely be crude at best.

Neurocomputing. In recent years a new computer processing technology has emerged capable of human like performance in such intelligent information processing tasks as classification and pattern association. Known alternately as neural networks, neurocomputing, or parallel distributed processing, this technology may overcome many of the limitations of traditional mathematical tools.

Neural networks can solve problems involving complex mappings or relationships which do not lend themselves to conventional algorithmic solution. This characteristic suggests neural networks may have a useful application as a tool for predicting organizational performance.

In particular, the recent emergence of back-propagation networks as an alternative means of function approximation may make it possible to "extract" the human judgement from past organizational performance evaluations and use it to independently derive composite performance ratings from subsequently collected data.

Unlike statistical regression, which fits selected classes of fitting functions to the data, back-propagation networks "learn" the functional relationship from exposure to training input-output examples. These networks form their own internal representation of the function instead of conforming to a hypothesized model. This characteristic permits back-propagation networks to fit data to unknown functional forms which may account for the apparent ability of back-propagation to approximate functions better than statistical regression techniques (5:2). Neural networks have been used with impressive results in similar generalization problems. In particular, neural networks have been successful in scoring loan applications and predicting corporate bond ratings (6:40; 3:449).

This research applies this neurocomputing technology to the problem of predicting organizational performance. Specifically, the content and unit ratings of past civil engineering squadron unit effectiveness inspection reports will be used to "teach" a candidate neural network to discriminate between "excellent" and "satisfactory" organizations. The resulting trained network will be used to evaluate squadrons from separate report data.

This application demonstrates the utility of neurocomputing as a measurement and prediction tool for the organizational sciences and perhaps may lead ultimately to an organizational assessment tool for managers.

II. Background

History of Neurocomputing

Research in the area of neural networks began some forty years ago with the work of early cognitive scientists seeking to understand the mechanisms of biological neural functioning. Inspired by the physiology and impressive information processing ability of the human brain, this work eventually led to mathematical models and ultimately physical implementation of computer networks of neuron-like elements.

One of the earliest implementations of such networks was the single-layer perceptron developed by Rosenblatt in the late 1950s (18:153-159). A mathematical model of a single layer perceptron is included in appendix A. This network generated much interest when initially developed because of its ability to recognize simple patterns.

By 1969, however, Minsky and Papert proved single-layer perceptrons were incapable of solving complex pattern recognition problems with overlapping decision regions (18:111; 14:227-232). An example of such a problem is the exclusive-or parity problem described in appendix B. Widespread interest in neural networks subsequently waned and it was not until the recent work of Hopfield, Rumelhart and others that vigorous research in this area resumed (13:37).

Today neural network models have been developed for whole classes of problem specific applications. These include combinatorial optimization (e.g. The travelling salesman problem), pattern recognition (e.g. speech and image processing) and classification (e.g. predicting bond ratings) (8:141-152; 4:826-834; 3:443-450; 11:183-192).

Simple Pattern Recognition

Many neural network applications involve some form of pattern association - matching a set of inputs with a predetermined class of outputs. For example, a single layer perceptron can associate pixel pattern coordinates with certain typed characters.

A single layer perceptron uses a collection of one or more processing elements to receive and process input signals and transmit output signals (see appendix A). The input signals correspond to variables (pixel pattern coordinates) and the signal emerging from the network corresponds to a "mapped" classification (typed character).

Borrowing the mathematical function analogy, this network is like a 'black box' which receives input signals corresponding to independent variables. After 'turning the crank', the emerging signals represent the dependent variable or nominal classification.

The internal state of the network (connection weight values) is determined from exposure to input-output examples. The internal state of the network, in effect, 'captures' or defines the functional relationship between the input variables and nominal classes. The ability of single layer perceptrons to associate or "map" a set of inputs with a class of outputs is limited to simple linearly-separable classes (see appendix B).

To support more complicated mappings, multi-layer networks are required. Unlike a single-layer perceptron, the pathway from input to output in a multi-layer network goes through more than one processing element.

By the mid 1980s, a learning algorithm for multi-layer networks had been developed which could modify the internal state of hidden layers in response to supervised training. Known as back-propagation, this algorithm employs a gradient descent heuristic that enables a network to self-organize in ways that improve its performance over time (10:155). On-going research into the theory and application of back-propagation networks suggests such networks are "universal approximators" capable of inducing arbitrarily complex input-output mappings (5:2; 9:3; 10:160).

Multi-layer Perceptron

A multi-layer perceptron or feed-forward network, consists of a collection of processing elements arranged in multiple layers. Each processing element receives many input signals from elements in previous layers but transmits only a single output signal. This output signal fans out along many pathways to provide input signals to processing elements in subsequent layers (figure 1). Each processing element maintains in local memory the values of adaptive coefficients known as "weights". These weights correspond to connection strengths between processing elements (6:38).

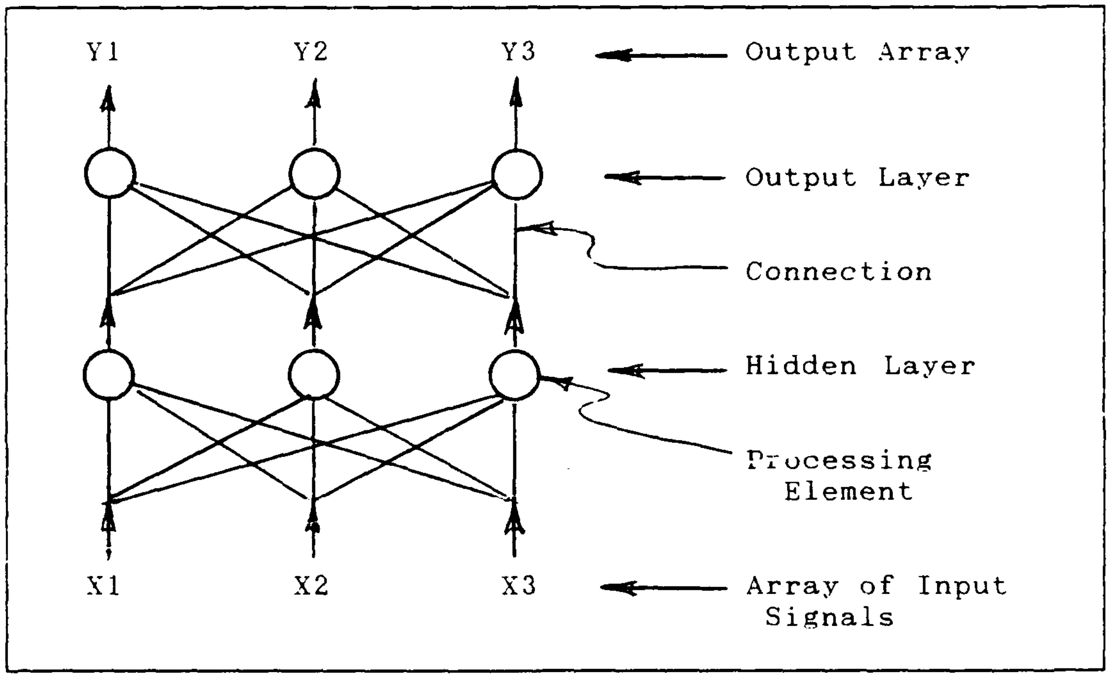
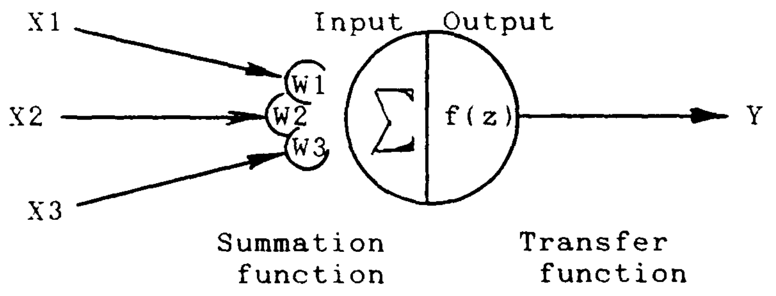


Figure 1. Multi-layer Perceptron.

The value of each element output signal is determined by a transfer function - a mathematical formula that defines the elements' output as a function of whatever input signals have just arrived and the weights present in local memory (6:38). Figure 2 is a mathematical model of a single processing element.



Xi = incoming signal
Wi = connection weight
Wo = bias similar in function to a threshold
z = net input signal = $\sum Wi*Xi + Wo$
Y = output signal = f(z) = f($\sum Wi*Xi + Wo$)
f(z) = transfer function

Figure 2. A Simulated Neuron

The transfer function is typically the sigmoid logistic function (figure 3), although it can be any differentiable function. The sigmoid function is defined as,

$$f(z) = 1/(1 + \exp(-z)).$$

When a vector array of input signals is imposed upon a trained network, processing elements in the first layer act simultaneously to process and transmit signals to the second layer. The second layer simultaneously processes and transmits signals to succeeding layers until the output signal array emerges from the network.

One way to regard such a multi-layered network is as a "black box". Entering values are real or binary signals representing indicator variables and the output is an array

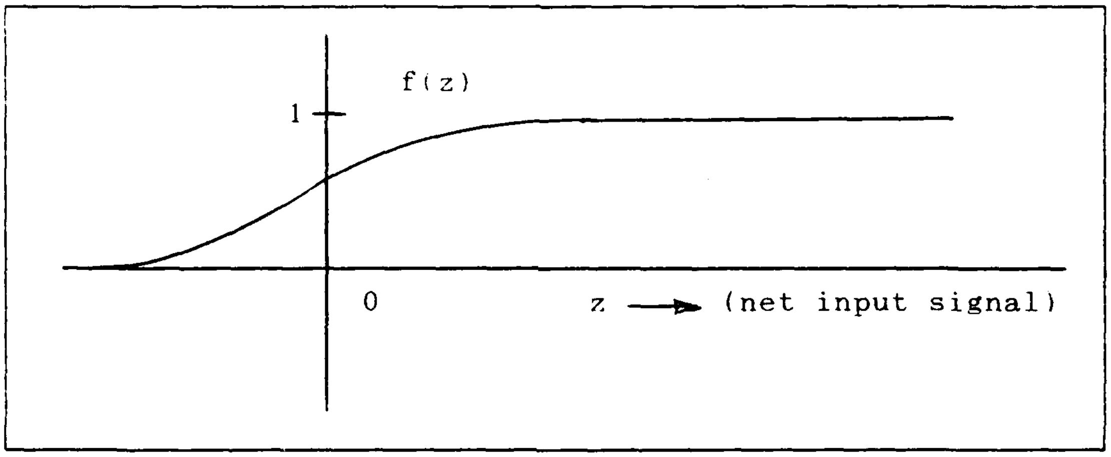


Figure 3. Sigmoid Function

of binary signals corresponding to a nominal-type classification. The hidden and output layers of processing elements represent the "black box" performing the functional mapping between predictor variables and nominal class.

Network learning

A multi-layer network learns how to perform the correct functional mapping by way of supervised training. In supervised training, the network is supplied with both input data and desired output data (correct answers as examples). In each trial, an input example is imposed on the network and the actual network output is compared with the desired output. If the output is in error, this error or "delta" is used to make corrections to elements in the hidden layers by the generalized delta rule (see appendix C).

The delta rule is a means of propagating the output error backward through the connections to previous layers. These distributed deltas are then used to modify the corresponding connection weights. This process is repeated in subsequent training trials until the weights converge to stable values and the output error reaches an acceptable level. By this method, the network adapts and organizes information within itself to "learn" the functional mapping or association. (6:38; 18:319-328; 10:158-160)

The back-propagation algorithm is a generalization of the least mean square algorithm using a gradient descent technique to minimize the mean square difference between the desired and actual net outputs (13:49). As such, this method may result in convergence to a local minimum instead of the desired global minimum error. Occurrence of local minima can be reduced or avoided by modifying network architecture (allowing extra processing elements, etc.) or making many training runs starting from different sets of random weights (13:50; 18:328-334). The back-propagation algorithm is summarized in appendix D.

Applications

Multi-layer networks using back-propagation are ideally suited for parallel computer architectures. But, in the absence of a parallel computer, neural networks are typically implemented as programmed simulations on conventional serial-type computers (17:62; 6:17).

While some of these serial type networks may be awkward and computationally inefficient when implemented as programmed simulations, these network models have nevertheless been successfully applied to generalization problems where conventional mathematical modeling techniques have yielded poor results.

Neural networks have been shown to perform much better than statistical regression for applications such as predicting the default risk of bonds and evaluating the credit risk of bank loans (3:450; 6:17). Conventional regression approximations attempt to fit the data to a specified class of curve fitting functions. These methods assume the underlying relationship between predictor variables and the data follows this function. Judging from the complexity of these applications, such an assumption is likely to limit the utility of regression as a prediction tool. Neural networks, by contrast, learn their own internal representation of the underlying function and do not require any up-front assumptions. In the bond rating application, a 10 variable neural network model has demonstrated an 88% success rate in correctly classifying bond ratings versus 65% for the linear regression model (3:449).

The problem of assessing organizational effectiveness is a complex one typically requiring the application of expert judgment to recognize patterns corresponding to a given level of performance. Variables influencing organizational effectiveness may involve complex second and third degree interaction and are typically large in number relative to the number of available data points. Hence, conventional statistical regression has not proven particularly useful as an analytical means of predicting organizational performance. (12:529; 7:1).

Neural networks have been successfully applied to similar generalization problems where conventional least squared estimation methods have yielded poor results (e.g. bond rating). Such problems share the difficulties inherent in an organizational performance assessment application; namely, a lack of knowledge about the underlying form of the functional relationships. The success of neurocomputing in applications where the underlying fuctional forms are unknown, suggests extension of this approach to the problem of predicting organizational performance may be a fertile area of application.

III. Method

Problem Summary

Civil engineering squadrons along with other base-level organizations undergo unit effectiveness inspections at periodic intervals. These inspections are conducted and documented by teams of experts dispatched from each major command headquarters. To ensure consistency among evaluations, each civil engineering squadron within a given command is evaluated by a common team of inspectors.

Because of the typical size and diversity of civil engineering squadrons, an evaluation team must consider and individually document the performance of a host of component branches, shops and other characteristics before a composite unit rating is applied. If the expert judgement used to combine these multiple observations into a single aggregate rating is applied consistently across all squadrons, it follows that an underlying pattern or deterministic relationship exists between documented observations and composite ordinal ratings. This research effort seeks to approximate this underlying relationship with a back-propagation neural network and to compare the prediction performance of the network with conventional regression models.

Summary of Method

This research will use the content and ratings of past unit effectiveness inspections to train a neural network to associate component organizational attributes with the correct ordinal performance rating. Inspection reports will be divided to provide data for both network training and testing. The training and testing samples will also be used to fit and test a regression model.

Collection of Data

Civil engineering unit effectiveness inspection reports from Strategic Air Command will be the source of the examples used for network training. The inspection reports are selected to fall within a finite period (approximately 24 months) and are limited to a single major command to minimize the influence of external factors which may diminish the accuracy of the network approximation. This limitation is designed to ensure uniform application of rating judgement and ensure relative homogeneity of civil engineering squadrons in terms of organization and mission without adversely restricting the sample size.

Processing Data

Each sampled inspection report will be examined. Any documented observation which may influence composite unit performance is identified as a candidate predictor variable.

A linear scale is employed to convert each identified observation into a value from -3 to +3 based on the report narrative associated with that observation. For example, if a report item suggests knowledge of communications security (COMSEC) is unsatisfactory based on the outcome of COMSEC tests, the variable corresponding to COMSEC would be assigned a value of -3. These identified variables form the input signals to the network. Each variable corresponds to an input node.

Network approximations are generally robust and tolerant of substantial error. Hence, any error introduced in the scoring of borderline cases is not expected to adversely affect the results. The criteria for scoring individual observations is provided in appendix E.

Network Training

The evaluation reports are randomly partitioned into a training set and a testing set. Each set provides a representative mix of each rating type. Input variable arrays and corresponding desired outputs from the training set are used to train a candidate network as described in chapter II. A variety of architectures and alteration of network parameters may be required to achieve learning convergence.

Validation and Comparison

Inspection reports from the testing set will be used to predict the performance of the corresponding squadrons using both the regression model and the trained neural network. Testing of each model on data partitioned from the original sample is necessary to get an indication of the respective model prediction accuracy.

A paired sample statistical test will determine the significance of any difference between the number of correct predictions of the regression model and the trained neural network. A prediction accuracy as good or better than that achieved by regression will establish the neural network as an alternative measurement and prediction tool for the organizational sciences and a potentially useful approach to organizational performance evaluation.

IV. Results

Data Collection

Twenty three civil engineering squadron unit effectiveness inspection reports were obtained from Headquarters Strategic Air Command. These reports documented inspections conducted between April 1987 and March 1989 and included 19 flying wing bases and 4 missile wing bases. Although possible unit ratings range from "unsatisfactory" to "outstanding", the reports obtained included squadron ratings of "marginal" (1), "satisfactory" (6), and "excellent" (16). Because of the dearth of outliers, the scope of this research was limited to discrimination between "excellent" and "satisfactory" squadrons.

Identification and Scoring of Variables

Each of the remaining 22 reports composing the sample of "excellent" and "satisfactory" squadrons were examined for component attributes which potentially influence the composite squadron performance rating. Thirty two candidate predictor variables were subsequently distilled from the collection of reports. Although some of these variables were not explicitly addressed in all reports, they nevertheless correspond to organizational components or functions common to all squadrons command-wide.

Identification and scoring of some variables was complicated by the on-going command-wide reorganization of operations branch shops to a zonal maintenance concept. Reports dating from early 1988 began reflecting changes in the organizational structure of the operations shops and required careful interpretation and scoring to maintain consistency. For example, the new operations shop organization included a "horizontal construction" section which performs functions formerly handled by the old "pavements and grounds" section. The performance of this function under both the old and new organization was consequently scored under the common variable "HORIZ".

Another complication arose in scoring of some borderline cases when the report narrative included both favorable and unfavorable comments. Such cases required subjective interpretation to determine the predominant "tone" of the narrative.

Assigning values to the candidate variables was otherwise fairly straight-forward. Scoring criteria and examples are included in appendix E.

To construct a regression model, there must be enough degrees of freedom available to estimate the mean squared error variance (19:11). Hence, with 22 cases available (21 with a cross-validation sample withheld), the number of variables in the model must be limited to 19 or less.

The original set of 32 variables was, therefore, reduced to 18 to permit construction and cross-validation of regression models.

Substantial reduction in the number of variables was achieved by consolidating some component functions into a single parent branch or function. For example, the variables corresponding to Design, Construction Management, Environmental & Contract Planning and Real Property were replaced by a single variable representing the parent branch, Engineering and Environmental Planning. The corresponding report narratives include an overall assessment of this parent branch.

Other variables were eliminated based on low correlation with the dependent variable and subjective considerations. For example, the variable representing vehicle operator care had a very low correlation with the dependent variable relative to the other predictors (0.02). This variable was also qualitatively judged less important as a predictor of overall performance.

Revision and consolidation of the variable set eliminated most of the missing observations. Variables in the revised set were either explicitly addressed in the report narratives or were judged "neutral" based upon the report context. The revised variable set is provided in Table I. The complete data set with reports and corresponding variable values are included in Appendix F.

Because the contents and ratings of specific reports are privileged, no base or squadron is identified. For purposes of this research, each report is identified by a number from 1 to 22. These numbers are arbitrary and do not correspond to any particular base or squadron nomenclature.

Table I. Revised Variable Set

Variable	Description
DPCWD	Disaster Prep\Chem Warfare Defense
TRNG	Unit Training Manager
ADMIN	Administration\Orderly Room
DEI	Industrial Engineering
DEU	Financial Manager
DEE	Engineering & Env Planning
DEF	Fire Department
DEH	Family Housing
SHELP	Self-Help Store
PBEEF	Readiness Management
WKCTL	Work Control
LOG	Logistics\Material Control
PLAN	Planning
HORIZ	Horizontal Construction
VERT	Vertical Construction
ZONE	Craftsmen\Zonal Maintenance
UTIL	Utilities
SANIT	Sanitation\Water & Waste

Partitioning of Samples

Conventional data splitting was not deemed practical for validation purposes because of the small number of "satisfactory" samples (6) and the lack of available degrees of freedom. Consequently, the cross-validation of split samples was accomplished in the manner of the PRESS statistic calculation (15:430). This is accomplished by withholding one observation from the sample and using the remaining observations to fit the regression model. This model is then used to predict the withheld observation. This procedure is repeated for the remaining observations selected for validation.

The neural network cross-validation is accomplished in exactly the same manner. The same observations are used to train the neural network which in-turn is used to predict the withheld observation.

Selection of Reduced Variable Set

To broaden the comparison of the regression and neural network models, alternative reduced variable combinations were selected for obtaining predictions from common input data. Initially, several stepwise variable selection methods were used; however, backward elimination, forward selection and stepwise methods all yielded different model compositions. Qualitative judgement was necessary to arrive at the reduced variable combinations in Table II. Results of the stepwise analyses are included in Appendix G.

To maintain consistency in the regression - neural network comparison, each of the reduced set variables were retained in all cross-validation regressions even though, in many cases, the regression statistics suggested not all beta coefficients were significant. Separate optimization procedures along with construction of separate network architectures would otherwise have to be performed for each of the nearly 70 cross-validation regressions. Fixing the composition of the regression and neural network models a priori does not diminish the validity of the model to model comparison. Any differences in model to model prediction accuracy are readily discerned.

Table II. Reduced Variable Sets

Full 18 Var Set	Reduced 13 Var Set	Reduced 8 Var Set
DPCWD	ADMIN	DEI
TRNG	DEI	DEE
ADMIN	DEU	DEF
DEI	DEE	DEH
DEU	DEF	PBEEF
DEE	DEH	VERT
DEF	PBEEF	UTIL
DEH	WKCTL	SANIT
SHELP	LOG	
PBEEF	HORIZ	
WKCTL	VERT	
LOG	UTIL	
PLAN	SANIT	
HORIZ		
VERT		
ZONE		
UTIL		
SANIT		

Regression Analysis

Since the data is limited to two nominal classes (excellent and satisfactory), the following binary outcomes were established for the dependent variable:

1 = Excellent

0 = Satisfactory

Subsequent cross-validation predictions were interpreted "excellent" for fitted values greater than 0.5 and "satisfactory" for fitted values less than 0.5.

The logistic transformation was used in each cross-validation regression to constrain the response function ($0 < \hat{Y} < 1$) and improve the overall fit. Figure 4 is an example of the logistic response function.

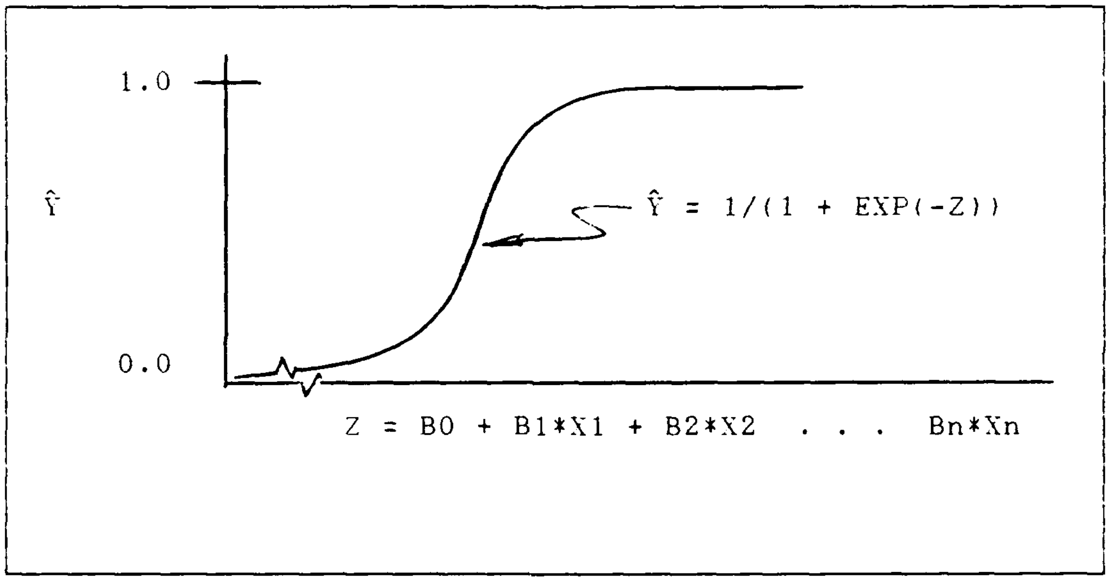


Figure 4. Logistic Response Function

Since unequal error variances are characteristic of models with dependent indicator variables, the logistic regression coefficients were computed using weighted least squares. Weighted least squares regression provides more efficient estimates and diminishes the effect of unequal error variances (15:354-367). Fitted values from initial unweighted logistic regressions were used to compute the weights. STATISTIX II analytical software package was used to compute the weighted logistic regression coefficients.

Neural Network Training

The same reports used to fit the cross-validation regression models were also used to train neural networks. Presentation of the training examples alternated between "excellent" and "satisfactory".

Two output nodes were used to represent the two nominal classes and the following desired training set outputs were established for each:

Y1	Y2	Rating
0	1	Excellent
1	0	Satisfactory

The trained network output node with the highest value is considered "on" and the other "off". Thus, the nominal class corresponding to the "on" node represents the network prediction.

Figure 5 is an example of an 8 variable network. In Figure 5, an array of values representing the 8 variables enter the network at the bottom. Emerging from the top are two signals, each corresponding to a nominal class ("satisfactory" or "excellent"). For a trained network, one signal is typically "large" (> 0.9) and the other "small"

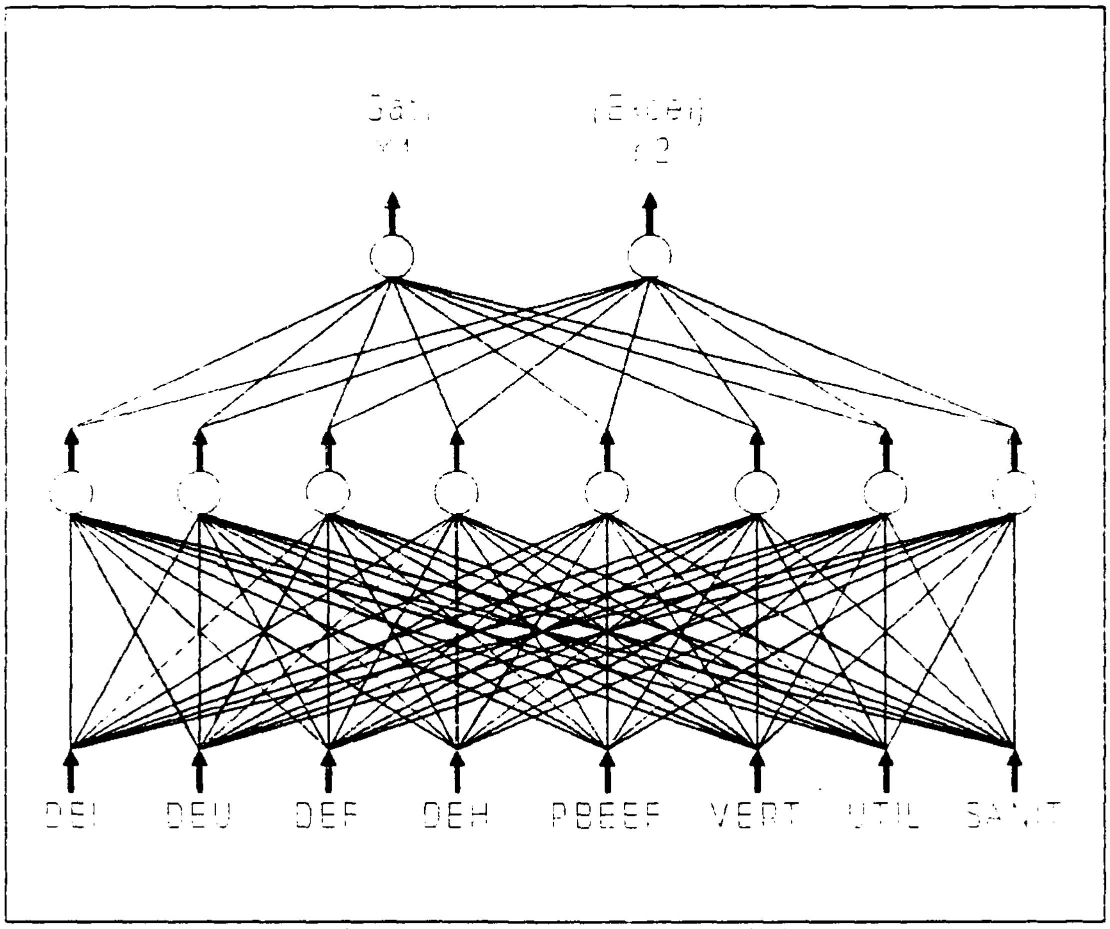


Figure 5. 8 Variable Neural Network Model

(< 0.1). The nominal class corresponding to the "large" signal represents the network prediction.

A Turbo Pascal neural network simulation written for this application is in appendix H. This program works but was superceded in favor of a machine language simulation package developed by Neural Ware Inc. The machine language simulation was used for its greater speed and computational efficiency.

Initial experimentation with different network architectures determined that a two layer network with N inputs and N hidden nodes was sufficient to achieve learning convergence. Adding nodes to the hidden layer reduces the required number of cycles to convergence but does not speed convergence in terms of processing time (limitation of software simulations). Adding a second hidden layer increases the required cycles to convergence. Evaluation of the different architectures on a small sample yielded identical prediction results. This is consistent with past experience in similar applications (3:449).

Figure 6 is a plot of the root mean squared error versus number of training cycles of different network architectures and variable compositions. Each of the networks used for the cross-validation trials converged within 200 presentations.

Results

The reports used for cross-validation predictions consisted of the six satisfactory reports and six randomly selected excellent reports. A prediction for each of these twelve reports was obtained by withholding each in-turn from the full sample and using the remaining reports to fit a regression model and train a neural network. Each of the twelve fitted models and trained networks were then used to predict the corresponding withheld report.

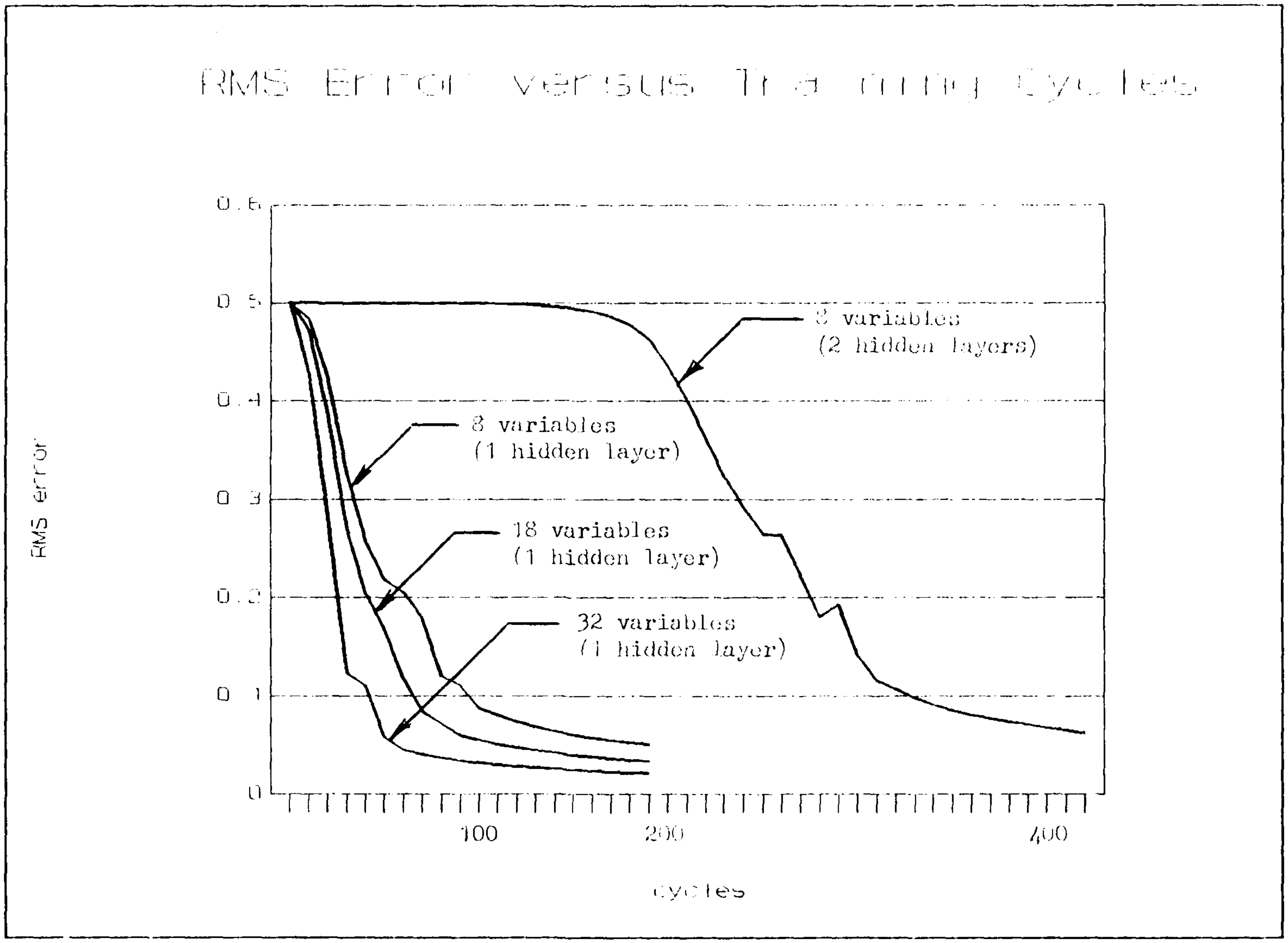


Figure 6. RMS Error versus Training Cycles

The full sample of reports used to fit the 8 variable, 13 variable, and 18 variable models are included in Table III. The reports selected for cross-validation are identified in Table IV. The full sample of reports were balanced to the extent permitted by the degrees of freedom (e.g. 6 Satisfactory and 6 Excellent reports for the 8 variable model).

Table III. Full Sample of Reports (For Regression and Network Training)

Model Composition	Training/Model Fitting Reports	
8 Variable	Excel:	8, 10, 4, 13, 20, 18
	Sat:	5, 9, 12, 17, 19, 22
13 Variable	Excel:	8, 10, 4, 13, 20, 18, 3, 7, 16, 21,
	Sat:	5, 9, 12, 17, 19, 22
18 Variable	Excel:	8, 10, 4, 13, 20, 18, 14, 1, 11, 15, 6, 2, 3, 7, 16, 21
	Sat:	5, 9, 12, 17, 19, 22

Table IV. Reports Selected for Cross-Validation Predictions

Excellent	Satisfactory
8	5
10	9
4	12
13	17
20	19
18	22

Twelve cross-validation predictions were obtained for each of the model compositions (8, 13, 18 variables). The raw output for the 8 variable regression and neural network predictions are summarized in table V. Appendix I is a complete summary of the regression and network predictions for the three different model compositions.

Table V. Raw Output (8 Variable Predictions)

	(Regression)		(Neural Network)			
	Pred	Actual	Predicted		Actual	
Report	Y	Y	Y1	Y2	Y1	Y2
8	1.0000	1	0.0627	0.9368	0	1
10	0.0337	1	0.1126	0.8885	0	1
4	1.0000	1	0.0402	0.9595	0	1
13	1.0000	1	0.2729	0.7291	0	1
20	1.0000	1	0.0305	0.9694	0	1
18	0.0173	1	0.4723	0.5267	0	1
5	0.0167	0	0.9920	0.0081	1	0
9	0.9921	0	0.1006	0.8999	1	0
12	0.0000	0	0.9936	0.0066	1	0
17	0.0000	0	0.8366	0.1644	1	0
19	0.0000	0	0.9920	0.0080	1	0
22	0.9729	0	0.0463	0.9535	1	0

Analysis of Data

Table VI summarizes the paired prediction outcomes of the regression - neural network cross-validations. The prediction of each withheld report is the product of a separate trained network and fitted regression model. This table includes the total number of correct predictions out of 12 predictions trials for each model composition.

Table VI. Prediction Outcome Summary

Test Pair	Ratio of Correct Pred.	PCT Correct
8 Variable		
Regression 8 variables	8/12	67%
Neural Net 8 variables	10/12	83%
13 Variable		
Regression 13 variables	6/12	50%
Neural Net 13 variables	9/12	75%
18 Variable		
Regression 18 variables	8/12	67%
Neural Net 18 variables	10/12	83%

While the neural network outperforms the regression model in each test pair, a statistical test is necessary to determine if the apparent difference in performance is significant. Because of the small sample, a paired measures design was used for the cross-validations and a pairwise statistical comparison was performed.

Table VII is a summary of the individual prediction outcomes of the 8 variable test pair. Since the prediction outcomes are expressed in terms of binary measures

(1 = success, 0 = failure), it follows that the population of difference scores of the paired measures are not normally distributed. Therefore, non-parametric analysis must be performed to determine the significance of differences in the paired prediction outcomes.

Table VII. Prediction Outcomes (8 Variable Pair)

Report		(Prediction Outcomes) Regression	Network
8	exc	success (1)	success (1)
10	exc	failure (0)	success (1)
4	exc	success (1)	success (1)
13	exc	success (1)	success (1)
20	exc	success (1)	success (1)
18	exc	failure (0)	success (1)
5	sat	success (1)	success (1)
9	sat	failure (0)	failure (0)
12	sat	success (1)	success (1)
17	sat	success (1)	success (1)
19	sat	success (1)	success (1)
22	sat	failure (0)	failure (0)
TOTAL		8	10
MEAN		0.67	0.83

The sign test is a non-parametric alternative to the paired t-test which requires no distributional assumptions about the paired samples. To perform the sign test, the following null and research hypotheses are posited:

Ho: Net - Reg = 0 (Net = Reg)

Ha: Net - Reg > 0 (Net > Reg)

where "Net - Reg" is the mean difference of the paired measures.

Rejection of the null in favor of the research hypothesis suggests there is a significant difference in the prediction accuracies of the neural network and regression models. Rejection should be based upon the weight of evidence provided by the level of significance of each test.

STATISTIX II analytical software package was used to perform the sign test computations. The results of the sign test are summarized in Table VIII.

Table VIII. Sign Test Results

Test Pair	Alt Hyp	One Tail P-Value	Reject Null Hyp
8 var	Net > Reg	0.250	yes
13 var	Net > Reg	0.125	yes
18 var	Net > Reg	0.313	no

Although specific p-values are subject to interpretation, the weight of evidence provided by the test results supports the research hypothesis. Judging from non-parametric tests in general and these p-values in particular, the power of the tests are probably weak. If the paired outcomes could be expressed in terms of "N" successes or a percentage instead of binary outcomes, the assumption of normality could be evaluated and more powerful parametric analysis would be possible.

One way of creating a more powerful test is by combining the prediction outcomes. Since each cross-validation report was tested on three independently fitted models (8 variable, 13 variable, 18 variable), the three outcomes of each sampled report can be consolidated into a single observation expressed in terms of N successes out of M trials or a percentage. With the data in this consolidated form, parametric assumptions can be evaluated to determine if a paired t-test can be performed. Table IX is a summary of the consolidated outcomes of the cross-validation predictions.

Table IX. Consolidated Outcomes

Report	Regression	N. Network
8	1/3 (0.33)	3/3 (1.00)
10	1/3 (0.33)	3/3 (1.00)
4	3/3 (1.00)	3/3 (1.00)
13	2/3 (0.67)	3/3 (1.00)
20	3/3 (1.00)	3/3 (1.00)
18	1/3 (0.33)	3/3 (1.00)
5	3/3 (1.00)	3/3 (1.00)
9	0/3 (0.00)	0/3 (0.00)
12	2/3 (0.67)	3/3 (1.00)
17	2/3 (0.67)	2/3 (0.67)
19	3/3 (1.00)	3/3 (1.00)
22	1/3 (0.33)	0/3 (0.00)
MEAN	0.611	0.806
VARIANCE	0.118	0.151

Evaluation of parametric assumptions suggests a paired t-test is appropriate. Judging from the paired sample variances, the assumption of equal variance should not be rejected. Rankit plots of the sample difference scores indicate no serious departure from normality (Appendix J).

Based on these parametric assumptions, a paired t-test was performed using the null and research hypotheses posited previously. To hedge against invalid assumptions, the non-parametric signed-rank test was also performed. Table X summarizes the results.

Table X. Test Results (Consolidated Outcomes)

Alt Hyp	Paired t test One Tail P-Value	Signed Rank test One Tail P-Value
Net > Reg	0.0337	0.0469

Findings. Both parametric and non-parametric analysis of the consolidated paired outcomes provide strong support for the research hypothesis. The results indicate a significant difference between the neural network and regression performance. The p-values in Table X reflect less than five percent probability that the improvement of the neural network over the regression model (81% versus 61%) happened by chance.

Analysis of Alternate Sample Data

A pairwise comparison was repeated for an alternate selection of cross-validation reports. A different random selection of six Excellent reports were combined with the six Satisfactory reports and twelve additional cross-validation predictions were obtained for each model composition (8, 13, 18 variable). Outcomes of each of the model predictions were consolidated as in the original analysis. The consolidated outcomes are summarized in Table XI.

Table XI. Consolidated Outcomes (Alternate Sample)

Report	Regression	N. Network
14	1/3 (0.33)	2/3 (0.67)
1	2/3 (0.67)	1/3 (0.33)
11	3/3 (1.00)	3/3 (1.00)
15	1/3 (0.33)	2/3 (0.67)
6	3/3 (1.00)	1/3 (0.33)
2	1/3 (0.33)	1/3 (0.33)
5	3/3 (1.00)	3/3 (1.00)
9	0/3 (0.00)	0/3 (0.00)
12	1/3 (0.33)	2/3 (0.67)
17	3/3 (1.00)	2/3 (0.67)
19	3/3 (1.00)	1/3 (0.33)
22	1/3 (0.33)	0/3 (0.00)
MEAN	0.611	0.500
VARIANCE	0.138	0.111

Findings. Inspection of the composite prediction accuracies (Net = 50%, Reg = 61%) is sufficient to conclude that the evidence does not support the research hypothesis (Net > Reg) for this sample of reports.

Interpretation of Results

The inconsistency between the results of the original and alternate sample analysis is apparently due to anomalous sample to sample variation. Closer examination of the alternate sample reports reveals that the six Excellent reports (14, 1, 11, 15, 6, 2) are very close to the six Satisfactory reports (5, 9, 12, 17, 19, 22) in content if not output rating.

Based upon the alternate sample report narratives, it was very difficult to discriminate between Excellent and Satisfactory reports. This is confirmed by the lower correlation with the dependent variable and much lower adjusted R-Squared value (0.0966 in the alternate sample versus 0.2093 in the original sample).

When the reports are very nearly the same in content, (as in the alternate sample), the neural network and regression model both have trouble "deciding" which rating to assign. Hence, the rating probability approaches that of a coin toss, 50/50 Satisfactory/Excellent.

Unlike the original sample, about half of the neural network models trained from the alternate sample converged to sub-optimal local minima even after several attempts with different weight initializations. In other words, the RMS error did not reach the desired global minimum because the network converged to the wrong answer in one or more of the training examples. This suggests no meaningful discrimination can be made between the contents of the Excellent and Satisfactory reports of the alternate sample.

Randomization of samples is intended to diminish the effect of anomalous sample to sample variation. However, with small samples, such variation is still possible even with random partitioning. Because the Excellent and Satisfactory reports of the alternate sample are very close in content, the neural network and regression models are unable to discern any underlying function; hence, no meaningful discrimination between the two ordinal types can be made.

By contrast, there appears to be enough discriminating information in the content of the original sample reports that an underlying relationship can be approximated. The fact that the neural network was consistently better than 75 percent (81 percent composite) in correctly predicting performance ratings for the original sample models (8, 13, 18 variable) supports the assumption that a discernable relationship exists between the content and output ratings for this sample data.

Sources of Possible Error. Although the neural network was successful in approximating the functional relationship underlying the reports in the original sample, there are several potential sources of error in identification and scoring of report variables which could obscure the existence of a relationship between report content and output ratings.

Any reader of an IG report narrative, who is not intimately familiar with the unit inspected, will inevitably introduce some error into the scoring of variables, particularly since a discrete value must be applied in borderline cases.

The omission of external variables not specifically addressed in the reports could introduce an indirect confounding effect. For example, factors like host wing mission, gross facility square footage, or weather zone could significantly mitigate the effect of some variables which would otherwise diminish or enhance the composite rating.

The results of the alternate sample prediction outcomes, in particular, suggest that the composite squadron ratings are a function of more than just the component performances explicit in the report narratives. There are certain intangible qualities such as base appearance, work force morale and "customer satisfaction" which almost certainly enter into an evaluator's deliberations but are not explicitly documented in the inspection reports.

Evaluation of Method. Future refinements to the method used in this research should be directed toward reducing the opportunities for error in the identification and scoring of report variables.

The addition of more rating increments or a continuous scale could perhaps allow finer discrimination in cases where the report narrative suggests a borderline rating lies somewhere between existing rating increments.

Perhaps a better way to handle such borderline cases would be to score the variable low on the existing observation and then create a duplicate observation with the variable in question scored high.

External variables for which data is available should be included in the models (e.g. host wing mission, gross facility SF, weather zone). A variable corresponding to host wing mission was omitted in this case because all four of the missile wing bases in the sample were rated Excellent. Without a larger balanced sample of missile wing reports, such a variable would have been unduly weighted in favor of an overall Excellent rating obscuring the effect of the other predictors.

Many additional variables could have been extracted from the IG reports and used for neural network training, but the variables were not present in the majority of the narratives. For example, the refrigeration shop was explicitly addressed in only 6 of 22 narratives. Refrigeration shop was one of the variables eliminated because of its narrow input.

Implementation of the full set of 32 variables identified originally, yielded a neural network prediction accuracy of 67 percent during cross-validation. The diminished performance is attributed to the confounding effect of the large number of missing observations (114 missing observations) which had to be scored "neutral" on the corresponding variables.

V. Conclusions

Results

The successful application of any function approximation tool requires that there be an underlying relationship between predictors and the dependent variable. Where the existence of an underlying relationship could be established between report content and output ratings, the neural network significantly outperformed regression models in correctly predicting squadron performance during cross-validation trials (81% versus 61%). These results demonstrate that neurocomputing can be successfully applied in an organizational performance assessment application where the existing relationship between organizational components and composite performance is unknown.

The success of the neural network in discerning any relationship at all is quite remarkable given the relatively small sample size and opportunities for error in identification and scoring of variables.

Examination of the report contents and results of the analyses suggests any existing relationship is subtle and the function of additional extrinsic variables not identified in the data.

Neural Net versus Regression. The improvement in performance of the neural network over regression is attributed to the fact that regression is only valid where the general functional model is known. For a neurocomputing application, the functional relationship need not be known.

Unlike the neural network, which negotiates a complex error terrain in carrying out function approximations, the regression model approximation is analogous to a single ridge forming a boundary between "excellent" and "satisfactory" samples which is made steeper by the logistic transformation. For samples that do not lie squarely on either side of this ridge, random variation can position them "off-center" in the wrong direction, tipping the balance in favor of "excellent" or "satisfactory" regardless of the actual output rating.

The advantage of neurocomputing over regression should be even more apparent in broader applications involving more than two nominal classes. With sufficient training examples, a suitable network architecture can be employed to achieve the desired prediction accuracy.

By contrast, the utility of a regression model for applications involving multiple classes is limited. Because multiple classes must be expressed in terms of a single dependent variable, the precision of such an approximation can only diminish as the number of nominal classes increase.

Implications of Research

This research demonstrated the utility of neurocomputing as an alternative approach to performance measurement and prediction previously unavailable in the organizational sciences.

Prediction of organizational effectiveness is a complex problem involving subtle relationships among organizational components. Such relationships can only be crudely approximated, if at all, with conventional least square estimation methods. Neurocomputing provides a means to approximate these relationships and use them to independently predict the rated effectiveness of an organization.

Applications. With a definitive set of variables, appropriate scoring criteria, and sufficient training examples, this technology could have immediate application as a *management tool for squadron commanders.* A commander could use a trained network to gain insight into the composite effectiveness of his squadron relative to other squadrons command-wide. Such a tool would provide a means to gauge the effect of alternative decisions favoring conflicting squadron objectives or component functions. A commander could then allocate resources toward each objective or function in a way that maximizes overall unit effectiveness.

This technology could also provide a useful supplement to subjective evaluations conducted during unit effectiveness inspections. Neural network predictions could be used to establish "bench mark" or "baseline" ratings to help experienced evaluators gauge the consistency of their rating judgement, ensuring a fair and uniform application of rating judgement from squadron to squadron.

Inexperienced evaluators could also benefit from the collective "experience" of the neural network. This would be particularly useful as a means to help leaven out the erratic application of judgement which can be characteristic of an evaluation team in transition, undergoing radical changes in composition.

As new evaluations are performed, a neural network can be retrained, incorporating the new knowledge. This way, the "knowledge base" evolves with each new experience. This application of neurocomputing should be regarded as a supplemental tool since network predictions reflect the "bias" of previous inspections and do not consider other extrinsic and intangible factors.

In the organizational sciences, this approach to effectiveness measurement may prove to be a useful alternative to conventional least square estimation methods or theory- based mathematics (e.g. Data Envelopment Analysis). A trained network could be used to perform "sensitivity analysis" of the predictor variables. Input values can be varied or selected input nodes disabled to determine the relative influence of predictors. Such an analysis could provide some insight into the relative importance of variables within the control of the manager as well as external factors beyond his control.

Recommended Research

Follow-on research should be considered to determine the ultimate capability of neurocomputing as a prediction tool for organizational performance. Collection of evaluation data should be in sufficient quantities of more than two ordinal output ratings. A broader set of variables should be incorporated to include external factors for which data is available.

Future research with major commands could ask evaluation team members to identify appropriate variables which influence squadron performance. They could also develop criteria for scoring the variables. This should result in a more definitive set of variables and accurate interpretation of report narratives.

Sensitivity analysis should be considered in any future research. This would not only provide insight into the relative importance of specific variables but would also demonstrate the robustness or error tolerance of the network approximation.

There are other neurocomputing techniques different from back-propagation which could be explored in an organizational assessment application.

The Hopfield network has been used as a content-addressable memory. Such a network could be used to match an un-rated organization with the closest stored "exemplar" organization.

The Carpenter/Grossberg network has been used to segregate or "cluster" similar input patterns. This type of network could be used to explore where natural boundaries between ordinal ratings should lie.

Conclusion

Neurocomputing is a new, rapidly emerging technology that has only recently been explored in applications beyond signal, speech and image processing. This research demonstrates a successful application of neurocomputing as an organizational performance assessment tool.

In the organizational sciences, this approach to effectiveness measurement and prediction can be a useful alternative to least square estimation methods. Continued research will determine the ultimate capability of neurocomputing in practical management applications.

Appendix A: <u>Single Layer Perceptron</u>
(Ref. 13:45-47; 18:154-159)

A conventional multiple regression model uses beta coefficients to transform input values (predictor/indicator variables) into an output value (dependent variable). This gives rise to the 'black box' analogy where the regression function is regarded as a 'black box' that receives the input variables. After 'turning the crank' the value of the dependent variable emerges.

This same analogy applies to a perceptron. Signals corresponding to input variables enter the 'black box' and signals emerging correspond to a nominal classification. In this case the 'black box' consists of a layer of processing elements (Figure 7). Each processing element receives input signals and transmits an output signal. The path from each input signal to each processing element has a corresponding connection strength or "weight". The value of each processing element output signal is determined by a mathematical formula (Figure 8) that defines the element's output as a function of the input signals and corresponding connection weights. These connection weights can be regarded as little "beta coefficients" that evolve over time with repeated exposure to training input-output examples.

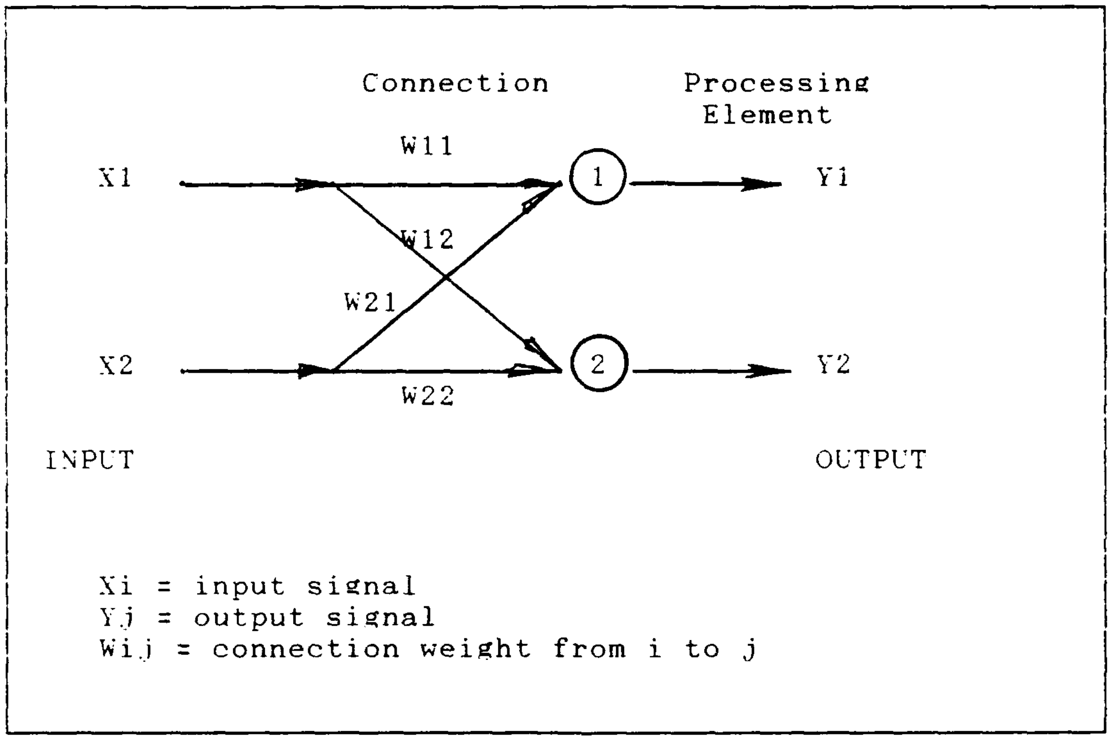


Figure 7. Single-layer Perceptron

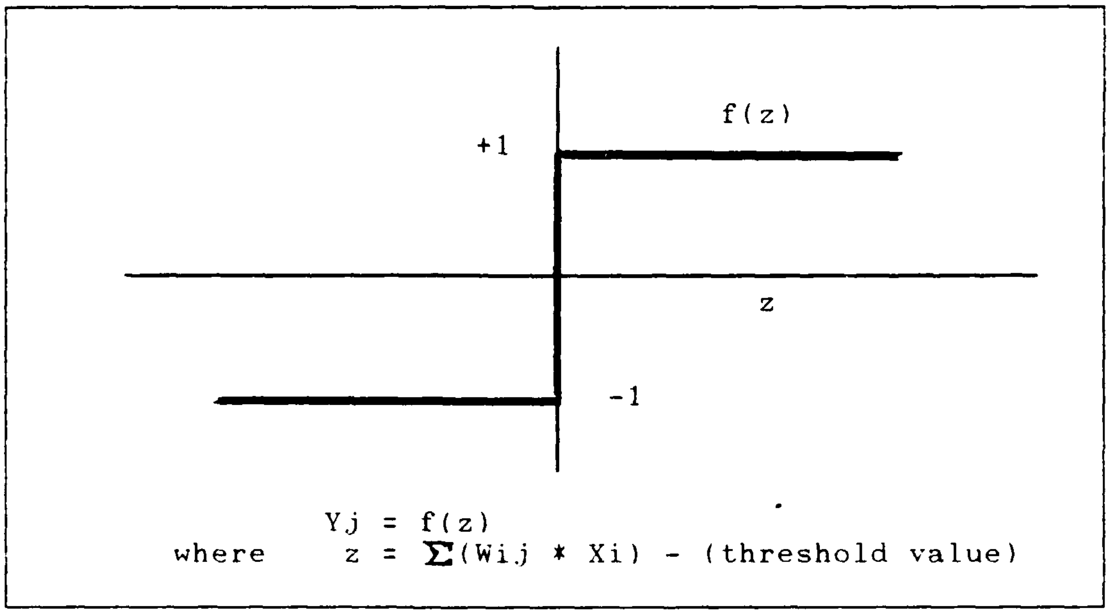
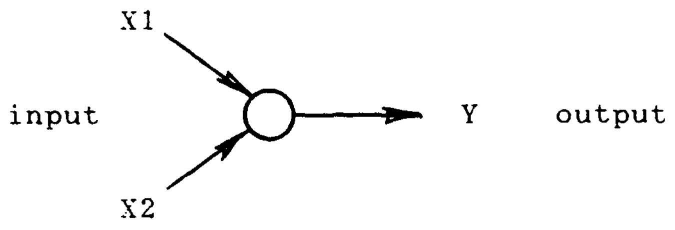


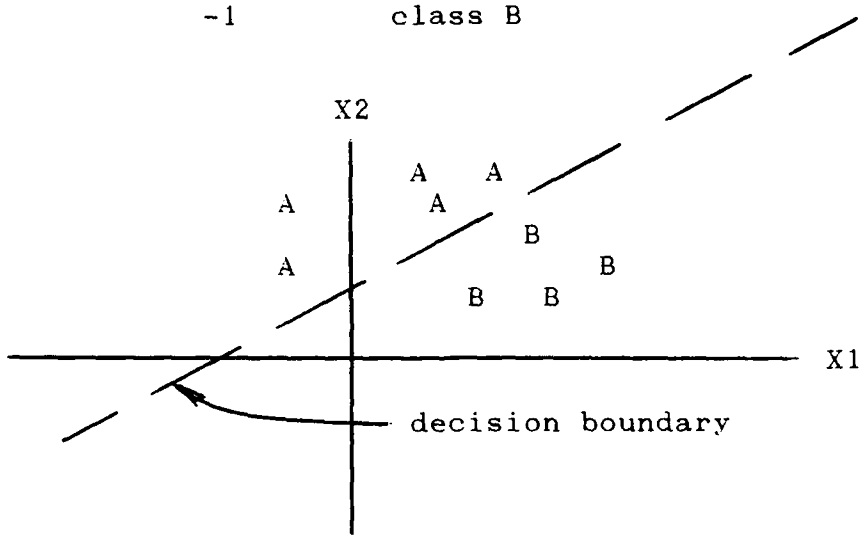
Figure 8. Hard Limiting Function

Appendix B: The Exclusive-Or Parity Problem and Linear Separability



Y = f(inputs, connection weights)

$$= \begin{cases} +1 & \text{class A} \\ -1 & \text{class B} \end{cases}$$



Shown above is a single layer perceptron that classifies an input vector (X1,X2) into two classes denoted A and B.

Figure 9. Perceptron Decision Region (from 13:45).

In performing classification tasks, single-layer perceptrons can only discriminate among linearly separable classes. In other words, in deciding whether an input belongs to one of two classes (denoted A or B), the perceptron forms two decision regions separated by a hyperplane in the space spanned by the inputs (figure 9).

The exclusive-or (XOR) parity problem is an example of a classification problem with meshed decision regions where the classes cannot be separated by a single hyperplane (or line in two dimensional space). The parity problem is one in which the output required is 1 if a binary input pattern contains an odd number of 1s and 0 otherwise. The XOR problem is a parity problem with input patterns of size two (13:45-46; 18:330-335).

Input pattern		Output pattern	
X1	X2	Y	Class
0	0	0	(B)
0	1	1	(A)
1	0	1	(A)
1	1	0	(B)

A single layer perceptron cannot solve this problem because the classes cannot be separated by a single line (figure 10).

The addition of multiple layers with hidden units allows the perceptron to form convex or closed regions in the space spanned by the inputs permitting separation of the classes. Thus, in the case of the XOR problem, a perceptron can learn to discriminate between the two classes (figure 11) via back-propagation.

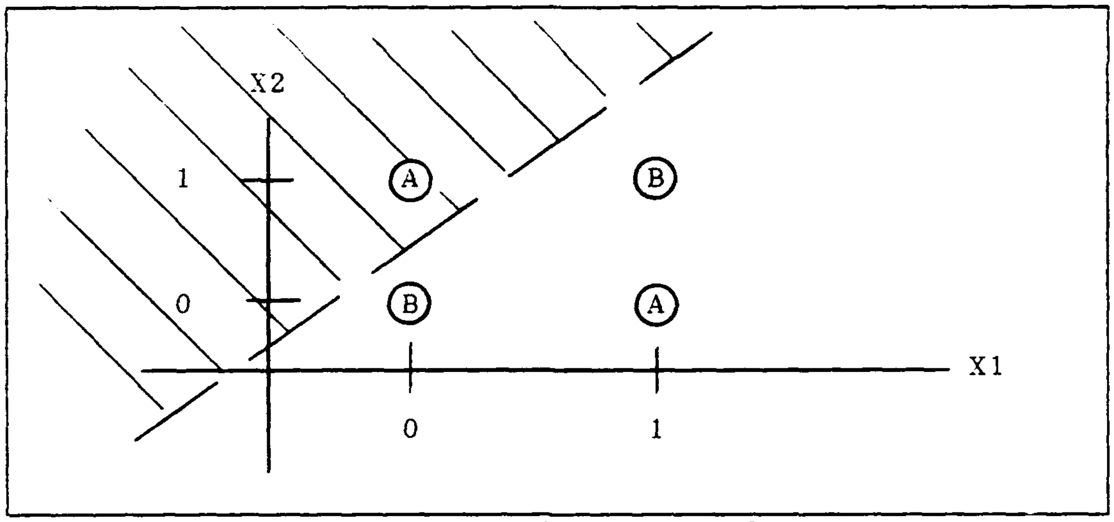


Figure 10. XOR Decision Region of Single-layer Perceptron (from 13:46).

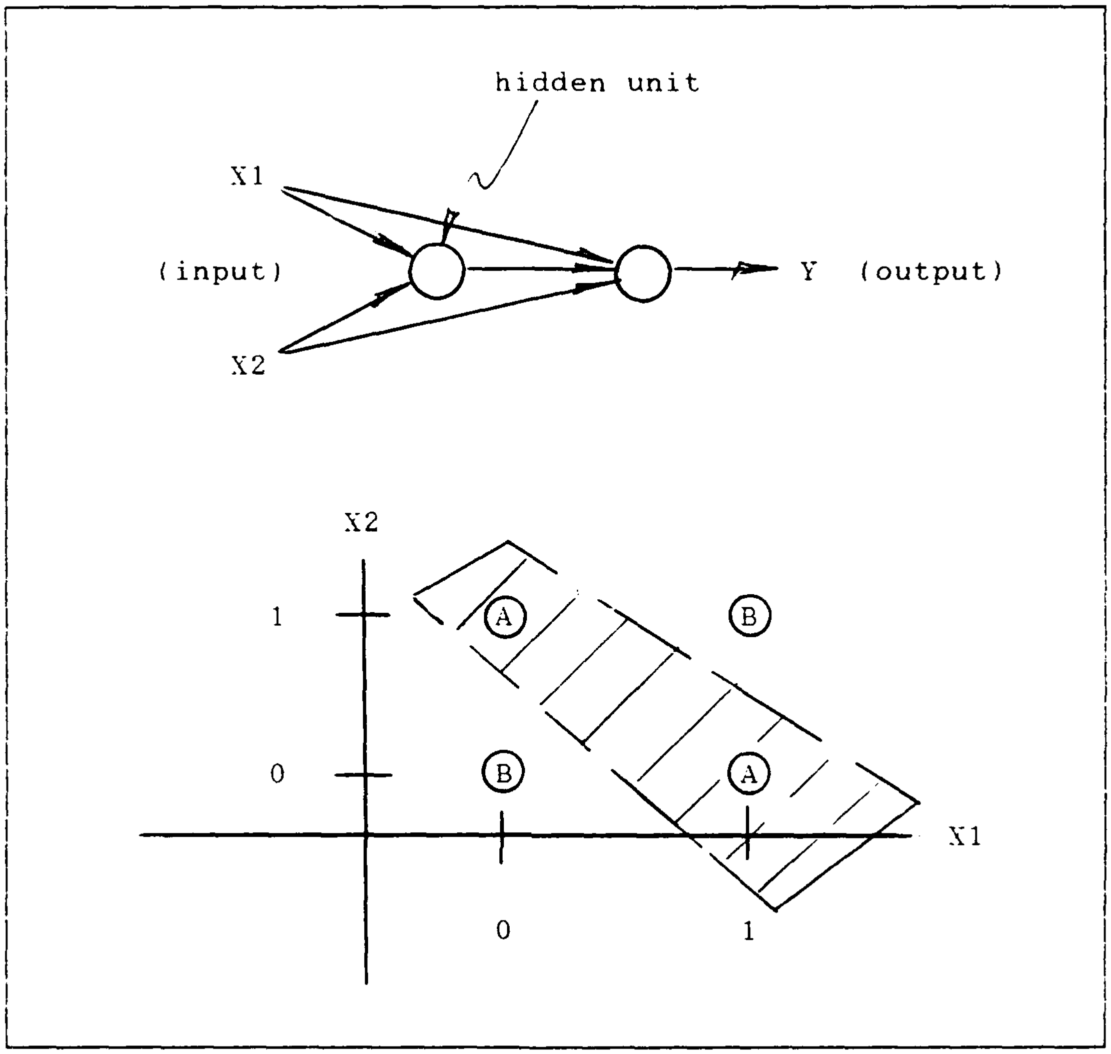
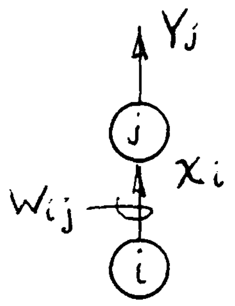


Figure 11. XOR Decision Region of Multi-layer Perceptron (from 13:46)

Appendix C: Generalized Delta Rule
(from 18:322-328)

The generalized delta rule is a means of incrementally modifying network connection weights to reduce the global output error during each training presentation. The generalized delta rule says the weight of each connection should be changed by an amount proportional to the product of the processing element output signal, Xi, and an error signal, δj, available to the unit receiving that output. In symbols,

Weight change = $\Delta Wij = \eta * \delta j * Xi$



where

η = gain term (learning rate)

δj = error signal downstream of j

Xi = output signal from unit i

Determination of the error signal starts with the output units. For an output unit, the error signal is given by

$\delta j = (Dj - Yj) * F'(z)$

where

Dj = desired output

Yj = actual output

z = net input signal

F'(z) = derivative of the transfer function

with the sigmoid function,

$F'(z) = Yj * (1 - Yj)$

thus

$$\delta j = (Dj - Yj) * Yj * (1 - Yj)$$

The error signal for hidden units is determined recursively in terms of the error signal of the units in the forward network layer to which it sends input and the corresponding connection weight. That is,

$$\delta j = F'(z) * \sum(\delta k * Wjk)$$

with the sigmoid function,

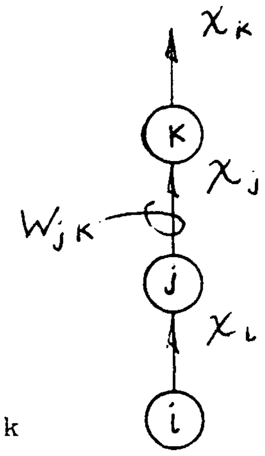
$$\delta j = Xj * (1 - Xj) * \sum(\delta k * Wjk)$$

where

Xj = output of unit j

δk = error signal in forward layer

Wjk = connection weight from unit j to k



This method of distributing error and updating weights is central to the back-propagation learning algorithm (see appendix D). Ref. (18:322-328) provides a good description and mathematical derivation of the generalized delta rule.

Appendix D: Back-Propagation Training Algorithm (from 13:49)

The back-propagation training algorithm is a gradient descent technique which employs the generalized delta rule to minimize the mean square error between the actual network output and the desired output (see appendix C). Each neuron (processing element) uses the sigmoid function.

Algorithm Steps:

Step 1. Initialize Weights and Threshold Values:

Set weights and thresholds to small random values for all nodes and all connections (e.g. between -0.1 and +0.1).

Step 2. Present Input and Desired Outputs:

From the training examples, present an input array representing the N input varia'les (X1, X2, X3, . . . Xn) along with the M desired output variables (D1, D2, D3 . . . Dm). The desired output array is specified by setting the output values to zero except for the output representing the desired classification which is set to 1. The M output units correspond to the M classes.

Step 3. Calculate Actual Outputs:

Using the sigmoid function, the output signals from the units in the first and succeeding layers are calculated in turn until the output array is determined (Y1, Y2, ... Ym).

Step 4. Adapt Weights:

Starting at the output layer and working back to previous layers in turn, the weights are adjusted as follows:

$Wij(t+1) = Wij(t) + \eta * \delta j * Xi'$

where $Wij(t)$ is the connection weight from unit i to unit j at time t, Xi' is the signal from unit i, η is a gain term and δj is an error term for unit j.

For output units,

$\delta j = (Dj - Yj) * Yj * (1 - Yj)$

where

Dj = desired output

Yj = actual output

For hidden units,

$\delta j = Xj' * (1 - Xj') * \sum(\delta k * Wjk)$

where $\sum(\delta k * Wjk)$ is summed over all K units in the layer downstream unit j.

Internal unit thresholds are adapted in a similar manner,

$Wj(t+1) = Wj(t) + \eta * \delta j$

where Wj is the internal threshold for unit j.

Step 5: Check for termination criteria.

If the RMS error is small enough (typically < 0.1) then stop training. Otherwise, repeat by going to step 2.

RMS Error = $SQRT(\sum_{1}^{n} (Di - Yi)^2 /n)$.

NOTE: The addition of a momentum term to the weight update can sometimes speed convergence and help avoid local minima.

Momentum term = alpha*(Wij(t) - Wij(t-1))

where

$0 < \text{alpha} < 1.$

Weight changes are thus smoothed as follows:

$$Wij(t+1) = Wij(t) + \eta * \delta j * Xi' + \text{alpha}*(Wij(t) - Wij(t-1)).$$

Appendix E: <u>Scoring Criteria</u>

Input variables are assigned values based on the following criteria:

<u>Outstanding (+3)</u>

Overwhelming balance of comments reflects singular performance. Includes adjectives such as "outstanding" or "superior".

Example, Variable DEH:

> "Family Housing Management continues to provide superior service to their customers. The level of effort and involvement is a model for the command . . . "

<u>Excellent (+2)</u>

Overwhelming balance of comments suggests performance beyond satisfactory. Includes adjectives such as "superb", "excellent", "exceptional", "exemplary".

Example, Variable DEH:

> "The Family Housing Management Branch provides superb support to the wing. A professional staff is providing customers useful information, particularly in housing referral . . ."

Satisfactory (three subsets):

<u>Favorable (+1)</u>

Comments are favorable in tone. Contains few if any unfavorable observations. Any unfavorable observations are off-set by overwhelming balance of favorable comments.

Example, Variable DEH:

> "Personnel in the Military Family Housing Branch are providing efficient, courteous service to all customers . . ."

Neutral (0)

Not mentioned/Omitted from report, or consists primarily of declarative sentences which do not reflect performance beyond what is expected. Favorable comments are off-set by unfavorable ones.

Example, Variable DEH:

> "The Family Housing Management Branch provides adequate support to the base and other military families . . ."

Unfavorable (-1)

Balance of comments are unfavorable in tone up to but not including marginal or unsatisfactory observations.

Example, Variable DEH:

> "The Family Housing Management Branch provides satisfactory support to the wing. The white exterior of the housing office is aesthetically pleasing, but the interior floor plan is not conducive to customer service . . . Base housing area appearance is suffering and a means of correcting this situation is underutilized."

Marginal (-2)

Comments are unfavorable in tone to include descriptions of "marginal" performance.

Example, Variable DEH:

> "The Housing branch is marginal for sustaining excessive housing unit downtime, failure to ensure housing units receive effective maintenance, and continuing a trend of poor area appearance . . ."

Unsatisfactory (-3)

Comments are unfavorable in tone to include descriptions of "unsatisfactory" conditions.

Example, Variable WKCTL:

> "Numerous discrepancies were noted within production control . . . the job order delinquency rate is 36 percent. THIS IS AN UNSATISFACTORY CONDITION . . ."

Appendix F: Data Set (18 Variables)

REPORT	Y1	Y2	DPCWD	TRNG	ADMIN
1	0	1	0	1	0
2	0	1	-1	0	2
3	0	1	-1	0	-1
4	0	1	0	0	2
5	1	0	1	0	0
6	0	1	-1	2	2
7	0	1	0	-1	1
8	0	1	-1	-1	1
9	1	0	-3	0	2
10	0	1	0	0	1
11	0	1	0	0	2
12	1	0	-1	2	2
13	0	1	0	0	2
14	0	1	1	0	0
15	0	1	0	0	0
16	0	1	1	-1	2
17	1	0	0	1	-2
18	0	1	0	1	2
19	1	0	-1	0	-2
20	0	1	0	0	-1
21	0	1	-1	-2	-2
22	1	0	0	0	1

REPORT	DEI	DEU	DEE	DEF	DEH
1	2	0	2	2	2
2	-1	1	1	3	1
3	1	2	2	2	3
4	1	3	2	2	1
5	0	2	-1	2	0
6	0	2	1	2	2
7	1	1	2	2	2
8	1	2	3	2	0
9	1	0	0	2	-1
10	0	3	2	2	0
11	1	1	2	2	2
12	-1	-1	1	2	2
13	1	0	2	2	2
14	1	1	-1	2	2
15	1	1	2	2	2
16	1	2	2	2	3
17	-1	0	1	-2	2
18	1	2	0	2	1
19	0	3	1	2	2
20	2	2	2	2	2
21	1	3	2	3	2
22	-1	3	1	2	2

REPORT	SHELP	PBEEF	WKCTL	LOG	PLAN
1	3	-1	0	0	2
2	0	1	1	1	1
3	-2	0	3	3	0
4	3	2	-2	0	0
5	2	-1	0	1	0
6	0	0	0	2	2
7	-1	2	2	2	2
8	0	1	1	-1	1
9	2	-1	1	2	1
10	2	2	1	1	0
11	1	1	2	2	0
12	-1	-1	1	0	1
13	3	1	0	2	1
14	2	0	1	1	0
15	0	1	0	1	1
16	2	0	1	1	2
17	0	0	1	1	1
18	3	2	1	1	0
19	0	-1	-1	3	2
20	1	3	1	0	0
21	3	2	0	1	1
22	0	2	1	1	0

REPORT	HORIZ	VERT	ZONE	UTIL	SANIT
1	0	2	0	0	3
2	0	1	0	1	0
3	2	1	0	0	0
4	-1	1	0	2	2
5	0	1	0	0	1
6	0	1	0	2	1
7	-1	3	0	2	2
8	1	1	0	2	1
9	1	2	0	1	1
10	1	2	1	2	2
11	0	1	1	2	2
12	-1	1	-1	0	1
13	0	1	0	1	3
14	1	1	0	1	1
15	0	1	0	2	3
16	1	2	0	2	1
17	1	2	2	1	1
18	1	0	1	2	1
19	1	0	0	1	0
20	0	1	1	1	2
21	1	1	1	2	0
22	0	0	0	2	1

Appendix G: Variable Selection Analysis

1. SAS output for Stepwise Regression Procedure:

```
                    STATISTICS FOR ENTRY:  STEP 3
                              DF  =  1,18

                                    MODEL
VARIABLE     TOLERANCE              R**2        F          PROB>F

DPCWD        0.954122              0.5335     0.4597      0.5064
TRNG         0.775372              0.5261     0.1690      0.6858
ADMIN        0.99064               0.5643     1.7629      0.2009
DEU          0.779936              0.5232     0.0599      0.8095
DEE          0.777264              0.5564     1.4095      0.2506
DEF          0.901046              0.5529     1.2597      0.2765
DEH          0.994093              0.5554     1.3657      0.2578
SHELP        0.816171              0.5250     0.1278      0.7249
WKCTL        0.995276              0.5294     0.2975      0.5921
LOG          0.94422               0.5218     0.0083      0.9282
PLAN         0.833871              0.5458     0.9592      0.3404
HORIZ        0.951211              0.5229     0.0482      0.8288
VERT         0.941929              0.5296     0.3062      0.5868
ZONE         0.825252              0.5325     0.4187      0.5257
UTIL         0.57931               0.5347     0.5065      0.4858
SANIT        0.780771              0.5216     0.0000      0.9999

NO OTHER VARIABLES MET THE 0.1000 SIGNIFICANCE LEVEL FOR ENTRY

SUMMARY  OF  STEPWISE  REGRESSION  PROCEDURE  FOR  DEPENDENT
VARIABLE Y2

            VARIABLE           NUMBER     PARTIAL    MODEL
STEP    ENTERED    REMOVED        IN         R**2      R**2       C(P)

   1    DEI                        1      0.3650    0.3650       7.51
   2    PBEEF                      2      0.1566    0.5216       3.22


            VARIABLE
STEP    ENTERED    REMOVED          F             PROB>F

   1    DEI                    11.4969           0.0029
   2    PBEEF                  6.22000           0.0220
```

2. SAS Output for Backward Elimination Procedure:

```
BACKWARD ELIMINATION PROCEDURE FOR DEPENDENT VARIABLE Y2

                  STATISTICS FOR REMOVAL:  STEP 11
                            DF = 1,13

                               PARTIAL   MODEL
                  VARIABLE       R**2     R**2

                  DPCWD         0.0851   0.6789
                  TRNG          0.0921   0.6719
                  DEI           0.0530   0.7111
                  DEE           0.0709   0.6931
                  DEF           0.1576   0.6064
                  PBEEF         0.1026   0.6615
                  HORIZ         0.0834   0.6806
                  VERT          0.0599   0.7041

ALL VARIABLES IN THE MODEL ARE SIGNIFICANT AT THE 0.10 LEVEL

SUMMARY  OF  BACKWARD  ELIMINATION  PROCEDURE  FOR  DEPENDENT
VARIABLE Y2


      VARIABLE  NO.  PARTIAL    MODEL
STEP  REMOVED   IN     R**2      R**2     C(P)      F

  1   ADMIN     17   0.0008    0.9245    17.03     0.0319
  2   ZONE      16   0.0017    0.9228    15.1      0.0909
  3   DEH       15   0.0152    0.9077    13.7      0.9822
  4   LOG       14   0.0172    0.8904    12.4      1.1183
  5   PLAN      13   0.0162    0.8742    11.05     1.0371
  6   SHELP     12   0.0166    0.8576    9.72      1.0564
  7   WKCTL     11   0.0270    0.8306    8.80      1.7061
  8   SANIT     10   0.0287    0.8020    7.96      1.697
  9   DEU       9    0.0237    0.7783    6.91      1.3152
 10   UTIL      8    0.0142    0.7640    5.48      0.7709
```

3. SAS Output for Forward Selection Procedure:

```
                  STATISTICS FOR ENTRY:  STEP 11
                            DF = 1,10

                                 MODEL
VARIABLE     TOLERANCE            R**2          F              PROB>F

DEU          0.403758            0.7878     0.2292             0.6424
SHELP        0.542032            0.7830     0.0005             0.9819
WKCTL        0.586084            0.7030     0.5307             0.4830
LOG          0.613619            0.7834     0.0198             0.8908
PLAN         0.476947            0.7880     0.2365             0.6373
ZONE         0.264332            0.7877     0.2212             0.6482
UTIL         0.414095            0.7892     0.2949             0.5990
SANIT        0.258208            0.7895     0.3090             0.5905

NO OTHER VARIABLES MET THE 0.45 SIGNIFICANCE LEVEL FOR ENTRY

SUMMARY OF FORWARD SELECTION PROCEDURE FOR DEPENDENT
VARIABLE Y2

       VAR      NO.   PARTIAL    MODEL
STEP   ENTERED  IN    R**2        R**2       C(P)       F

  1    DEI       1    0.3650     0.3650     7.51       11.50
  2    PBEEF     2    0.1566     0.5216     3.22       6.22
  3    ADMIN     3    0.0427     0.5643     3.50       1.76
  4    DEH       4    0.0628     0.6271     2.98       2.86
  5    HORIZ     5    0.0299     0.6570     3.77       1.40
  6    DEE       6    0.0173     0.6743     5.08       0.80
  7    DEF       7    0.0175     0.6919     6.38       0.80
  8    DPCWD     8    0.0259     0.7177     7.34       1.19
  9    VERT      9    0.0198     0.7375     8.55       0.90
 10    TRNG     10    0.0455     0.7830     8.72       2.31
```

Appendix H: Neural Network Simulation Program

```
program cenet18x(results, zerowt, train, error);

{
 Pascal program to implement back-propagation neural network
 learning.  Program simulates a two layer perceptron  with
 18 inputs, 18 hidden nodes and 2 outputs.  Program includes
 random or sequential presentation of training examples and
 a momentum term in the weight updates.

  Written by F. Baugh,  June 1989
}

  var
   z,        { weighted sums of inputs plus threshold value }
   eta,      { gain term (learning rate) }
   alpha,    { momentum coefficient }
   count,    { number of iteration }
   total,    { total iterations }
   step,     { iterations per stepped output }
   dsum,     {                        }
   sum,      {     place              }
   old,      {     holders            }
   err,      {                        }
   err1,     { error output node 1 }
   err2,     { error output node 2 }
   maxerr,   { maximum squared error }
   sumerr,   { cumulative squared error }
   maxroot,  { maximum root squared error }
   rms,      { root mean squared error }
   cycle,
   res:
      real;
   s,
   example,
   i,
   j,
   k,
   p,
   n:
      integer;
   l,
   m,
   q:
      char;
   test,
   train,          { training set data file }
   zerowt,         { initial weights data file }
   error,          { RMS error output file }
   results:        { "trained weights" output file }
      text;
   w: array[19..38]             { threshold weights }
        of real;
   w1: array[1..18,19..36]      { layer 1 connection weights }
        of real;
   w2: array[19..36,37..38]     { layer 2 connection weights }
        of real;
   delt: array[19..38]          {threshold momentum wt change}
         of real;
   delt1: array[1..18,19..36] { layer 1 momentum wt change }
        of real;
   delt2: array[19..36,37..38]{ layer 2 momentum wt change }
        of real;
   x: array[19..38]             { output, node j }
        of real;
   d: array[19..38]             { error term, node j }
        of real;
   yt: array[1..50,1..2]       { output 'y', training set }
        of real;
   xt: array[1..50,1..18]       { input 'x', training set }
        of real;
   yout: array[1..50,1..2]      { prediction, trained network}
        of real;                { yout = f(x1, x2,..., x18) }

 { input/output arrays are dimensioned to accomodate up to
 50 examples.  Dimensions may be changed to include more
 examples if needed.}

   label start;
   label stop;

BEGIN
clrscr;
{ message }
writeln('This program simulates a two layer perceptron with
18 inputs, ');
writeln('18 hidden nodes, and 2 outputs.  Network learning
includes  ');
writeln('cyclic presentation of inputs and momentum. ');
writeln(' ');
writeln('last revision: 1800, 4 Aug 89 ');
writeln(' ');
writeln('press <RETURN> to continue');
readln(q);


{ initializing parameters }

   assign(results,'outwt18.3');
   assign(zerowt,'zerowt.a1');
   assign(train,'train18.dat');
   assign(error,'error3.dat');
   rewrite(results);
   rewrite(error);
   reset(zerowt);
   reset(train);
   eta := 0.9;
   alpha := 0.6;
   count := 0;
   l := 'r';
   n := 1;


{ initialize weights }

{ reading threshold weights (nodes 19 to 38) }
j := 18;
while j < 38 do
      begin
      j := j + 1;
      readln(zerowt,w[j]);
      delt[j] := 0;
      writeln('Initializing threshold weights ',j);
      clrscr;
      end;

{ reading layer 1 connection weights }
i := 0;
while i < 18 do
      begin
      i := i+1;
      j := 18;
      while j < 36 do
            begin
            j := j+1;
            readln(zerowt,w1[i,j]);
            delt1[i,j] := 0;
            writeln('Initializing layer 1 connection weights
',j);
            clrscr;
            end;
      end;

{ reading layer 2 weights }
i := 18;
while i < 36 do
      begin
      i := i + 1;
      j := 36;
      while j < 38 do
            begin
            j := j+1;
            readln(zerowt,w2[i,j]);
            delt2[i,j] := 0;
            writeln('Initializing layer 2 connection weights
',j);
            clrscr;
```

```
                end;
          end;


    { reading training data }

    i := 0;
    while i < 50 do      { 50 = max number of training examples }
          begin
          i := i+1;
          read(train, yt[i,1], yt[i,2], xt[i,1], xt[i,2],
    xt[i,3], xt[i,4], xt[i,5], xt[i,6], xt[i,7], xt[i,8],
    xt[i,9], xt[i,10], xt[i,11], xt[i,12], xt[i,13], xt[i,14],
    xt[i,15], xt[i,16], xt[i,17], xt[i,18]);
          sum := yt[i,1]+yt[i,2];
              if sum = 1
                 then
                     example := i;
          writeln('reading training set ',i);
          clrscr;
          end;


    start:
    clrscr;
    writeln(' ');
    writeln('Data set includes ',example,' examples');
    writeln(' ');
    writeln('***** Network Menu *****');
    writeln(' ');
    writeln('          Options:');
    writeln(' ');
    writeln('          To implement network learning, press
    <RETURN>');
    writeln(' ');
    writeln('          To test the net with existing weights,
    enter "t",<RETURN>');
    writeln(' ');
    writeln('          To save weights enter "save",<RETURN>');
    writeln(' ');
    writeln('          To modify learning parameters enter
    "m",<RETURN>');
    writeln(' ');
    writeln('          To stop, enter "exit",<RETURN>');
    writeln(' ');

    readln(q);
    if q = 'e'
       then
           goto stop;
```

```
if q = 'm'
   then
       begin
       clrscr;
       writeln(' ');
       writeln('
Current');
       writeln('Option        Description
setting    ');
       writeln(' ');
       writeln(' 1             learning mode    >         ',l);

       writeln('               (r = random   )
');
       writeln('               (c = cyclical)
');
       writeln(' ');
       writeln(' ');
       writeln(' 2             learning rate    >
',eta:2:2);
       writeln(' ');
       writeln(' ');
       writeln(' 3             momentum coeff   >
',alpha:2:2);
       writeln(' ');
       readln(p);
       if p = 1
          then
              begin
              write('enter desired learning mode: ');
              readln(l);
              writeln(' ');
              if l = 'c'
                 then
                     writeln('presentation of training
examples will be cyclical')
                 else
                     writeln('presentation of training
examples will be random');
              end;

       if p = 2
          then
              begin
              write('enter desired learning rate: ');
              readln(eta);
              writeln(' ');
              writeln('learning rate = ',eta:2:2);
              end;
```

```
        if p = 3
           then
               begin
               write('enter desired momentum coefficient: ');

               readln(alpha);
               writeln(' ');
               writeln('learning rate = ',alpha:2:2);
               end;
        goto start;
        end;

   if q = 's'
      then
           begin
           { print trained weights to file }
           writeln('saving weights');

           { threshold weights }
           j := 18;
           while j < 38 do
                 begin
                 j := j + 1;
                 writeln(results,w[j]:4:4);
                 end;

           { layer 1 connection weights }
           i := 0;
           while i < 18 do
                 begin
                 i := i+1;
                 j := 18;
                 while j < 36 do
                       begin
                       j := j+1;
                       writeln(results,w1[i,j]:4:4);
                       end;
                 end;

           { layer 2 connection weights }
           i := 18;
           while i < 36 do
                 begin
                 i := i + 1;
                 j := 36;
                 while j < 38 do
                       begin
                       j := j+1;
                       writeln(results,w2[i,j]:4:4);
                       end;
                 end;
           goto start;
           end;
        { end weight save }


if q = 't'
   then
       begin

       { calculation of outputs }
       writeln('predicted output = f(x1, x2, x3, ... x18)');
       writeln(' ');
       writeln(' ');
       writeln('From input data:        Predicted:');
       writeln(' ');
       writeln('Y1      Y2               Y1      Y2 ');
       writeln(' ');


       { Forward calculation of neuron outputs for each
training set input vector }
       k := 0;
       while k < example do
             begin
             k := k+1;

             { layer 1 outputs }

             j := 18;
             while j < 36 do
                   begin
                   j := j+1;
                   i := 0;
                   sum := 0;
                   while i < 18 do
                         begin
                         i := i+1;
                         sum := sum + w1[i,j]*xt[k,i];
                         end;
                   z := sum + w[j];
                   x[j] := 1/(1 + exp(-z));
                   end;

             { layer 2 outputs }
             j := 36;
             while j < 38 do
                   begin
                   j := j+1;
                   i := 18;
                   sum := 0;
                   while i < 36 do
                         begin
                         i := i+1;
                         sum := sum + w2[i,j]*x[i];
```

```
                                  end;
                             z := sum + w[j];
                             x[j] := 1/(1 + exp(-z));
                             end;

                        writeln(yt[k,1]:4:0,'  ',yt[k,2]:4:0,'
 ',x[37]:4:4,'  ',x[38]:4:4);
                        end;
                  writeln(' ');
                  writeln('press <ENTER> to continue');
                  readln(q);
                  goto start;
                  end;


{ prompting for 'total' and 'step' }

write('enter total number of iterations: ');
readln(total);
writeln(' ');
write('enter number of iterations per output: ');
readln(step);
writeln(' ');
writeln('total = ',total:4:0);
writeln('step = ',step:4:0);
writeln(' ');
writeln(' ');
writeln(' ');
if l = 'c'
   then
      begin
      writeln('                    Training    RMS         RMAX');
      
      writeln('  Iteration    sweep      error
error');
      writeln(' ');
      end
   else
      begin
      writeln('                    RMS         RMAX');
      writeln('  Iteration    error       error');
      writeln(' ');
      end;


{ Training the Net }

while count < total do
      Begin
      count := count + 1;
      { begin presentation of training examples }

      if l = 'c'
         then
             begin
             k := n;
             if n = example + 1               { cyclic
selection }
                then
                    k := 1;
             n := n + 1;
             end
         else
             k := trunc(example*random) + 1;   { random
selection }


        { forward calculation of neuron outputs }

        { layer 1 outputs }
        j := 18;
        while j < 36 do
              begin
              j := j+1;
              i := 0;
              sum := 0;
              while i < 18 do
                    begin
                    i := i+1;
                    sum := sum + w1[i,j]*xt[k,i];
                    end;
              z := sum + w[j];
              x[j] := 1/(1 + exp(-z));
              end;

        { layer 2 outputs }
        j := 36;
        while j < 38 do
              begin
              j := j+1;
              i := 18;
              sum := 0;
              while i < 36 do
                    begin
                    i := i+1;
                    sum := sum + w2[i,j]*x[i];
                    end;
              z := sum + w[j],
              x[j] := 1/(1 + exp(-z));
              end;
        { adapting weights (backward error propagation) }

        { output error & threshold weight update }
        i := 0;
        j := 36;
        while j < 38 do
              begin
              j := j+1;
              i := i+1;
              old := w[j];
              d[j] := x[j]*(1-x[j])*(yt[k,i]-x[j]);
              w[j] := w[j] + eta*d[j] + alpha*delt[j];
              delt[j] := w[j] - old;
              end;

        { hidden layer error & threshold weight update }
        i := 18;
        while i < 36 do
              begin
              i := i+1;
              dsum := 0;
              j := 36;
              while j < 38 do
                    begin
                    j := j+1;
                    dsum := dsum + d[j]*w2[i,j];
                    end;
              old := w[i];
              d[i] := x[i]*(1 - x[i])*dsum;
              w[i] := w[i] + eta*d[i] + alpha*delt[i];
              delt[i] := w[i] - old;
              end;

        { hidden layer connection weight update }
        i := 18;
        while i < 36 do
              begin
              i := i+1;
              j := 36;
              while j < 38 do
                    begin
                    j := j+1;
                    old := w2[i,j];
                    w2[i,j] := w2[i,j] + eta*d[j]*x[i] +
   alpha*delt2[i,j];
                    delt2[i,j] := w2[i,j] - old;
                    end;
              end;
```

```
     { input layer connection weight update }
     i := 0;
     while i < 18 do
          begin
          i := i+1;
          j := 18;
          while j < 36 do
               begin
               j := j+1;
               old := w1[i,j];
               w1[i,j] := w1[i,j] + eta*d[j]*x[i] +
alpha*delt1[i,j];
               delt1[i,j] := w1[i,j] - old;
               end;
          end;


     { calculating outputs at each stepped pass }
     res := count/step;
     if frac(res) = 0
          then
              begin
              k := 0;
              maxerr := 0;
              sumerr := 0;
              while k < example do   { calculation of
                                       training set outputs }
                    begin
                    k := k+1;

                    { layer 1 outputs }

                    j := 18;
                    while j < 36 do
                          begin
                          j := j+1;
                          i := 0;
                          sum := 0;
                          while i < 18 do
                                begin
                                i := i+1;
                                sum := sum +
w1[i,j]*xt[k,i];
                                end;
                          z := sum + w[j];
                          x[j] := 1/(1 + exp(-z));
                          end;
                    { layer 2 outputs }

                    s := 0;
                    j := 36;
                    while j < 38 do
                          begin
                          s := s+1;
                          j := j+1;
                          i := 18;
                          sum := 0;
                          while i < 36 do
                                begin
                                i := i+1;
                                sum := sum + w2[i,j]*x[i];
                                end;
                          z := sum + w[j];
                          x[j] := 1/(1 + exp(-z));
                          yout[k,s] := x[j];
                          end;


                    { calculation of squared error }

                    err1 := sqr(yout[k,1] - yt[k,1]);
                    err2 := sqr(yout[k,2] - yt[k,2]);
                    if err1 > err2
                       then
                           err := err1
                       else
                           err := err2;

                    { maximum error }
                    if err > maxerr
                       then
                           maxerr := err;

                    { cumulative error }
                    sumerr := sumerr + err1 + err2;
                    end;
                  { end calculation of entire training set }


              { calculating RMS error at each stepped
                                                 iteration }
              cycle := trunc(count/example);
              rms := sqrt(sumerr/(example*2));
              maxroot := sqrt(maxerr);
              if l = 'c'
                 then
                     begin
                     writeln('   ', count:7:0, '     ',
cycle:7:0,'      ',rms:4:4,'      ', maxroot:4:4);
                     writeln(error, count:7:0, '     ',
cycle:7:0,'      ',rms:4:4,'      ', maxroot:4:4);
                          end
                      else
                          begin
                          writeln('    ',count:7:0, '
',rms:4:4,'      ', maxroot:4:4);
                          writeln(error, count:7:0, '
',rms:4:4,'      ', maxroot:4:4);
                          end;

                   end;
                   { end stepped iteration }

         End;
         { End network training }


        writeln(' ');
        writeln('Do you want to continue?');
        readln(q);
        if q = 'y'
           then
               goto start;

stop:
writeln('do you want to save weights ?');
readln(q);
if q = 'y'
     then
         begin
         { print trained weights to file }
         writeln('saving weights');

         { threshold weights }
         j := 18;
         while j < 38 do
               begin
               j := j + 1;
               writeln(results,w[j]:4:4);
               end;

         { layer 1 connection weights }
         i := 0;
         while i < 18 do
               begin
               i := i+1;
               j := 18;
               while j < 36 do
                     begin
                     j := j+1;
                     writeln(results,w1[i,j]:4:4);
                     end;
               end;
```

```
          { layer 2 connection weights }
          i := 18;
          while i < 36 do
                begin
                i := i + 1;
                j := 36;
                while j < 38 do
                      begin
                      j := j+1;
                      writeln(results,w2[i,j]:4:4);
                      end;
                end;
          end;
          { end weight save }

close(results);
close(zerowt);
close(train);
close(error);

END.

{ Frank Baugh }
```

Appendix I: Output from Cross-Validation Trials

8 Variable Model

Report	Actual Rating	(Neural Net Prediction) Y1	Y2	(Reg Pred) Y
8	exc	0.062662	0.936842	1.0000
10	exc	0.112565	0.888472	0.0337
4	exc	0.040213	0.959503	1.0000
13	exc	0.272907	0.729138	1.0000
20	exc	0.030481	0.969430	1.0000
18	exc	0.472339	0.526682	0.0173
5	sat	0.991958	0.008054	0.0167
9	sat	0.100630	0.899928	0.9921
12	sat	0.993596	0.006590	0.0000
17	sat	0.836645	0.164449	0.0000
19	sat	0.991960	0.007992	0.0000
22	sat	0.046344	0.953524	0.9729

13 Variable Model

Report	Actual Rating	(Neural Net Prediction) Y1	Y2	(Reg Pred) Y
8	exc	0.034160	0.967113	0.0000
10	exc	0.285186	0.715311	0.0141
4	exc	0.083628	0.917516	1.0000
13	exc	0.265724	0.740571	1.0000
20	exc	0.015948	0.984669	1.0000
18	exc	0.176932	0.825691	0.0559
5	sat	0.998800	0.001209	0.0000
9	sat	0.025167	0.974961	0.9949
12	sat	0.971944	0.027840	0.0000
17	sat	0.095357	0.905250	1.0000
19	sat	0.881802	0.120842	0.0000
22	sat	0.028366	0.972002	0.8586

18 Variable Model

Report	Actual Rating	(Neural Net Prediction) Y1	Y2	(Reg Pred) Y
8	exc	0.129772	0.869767	0.0006
10	exc	0.098736	0.902190	1.0000
4	exc	0.018796	0.981202	0.9994
13	exc	0.152416	0.847843	0.4522
20	exc	0.026731	0.973338	1.0000
18	exc	0.178046	0.820164	1.0000
5	sat	0.737891	0.269008	0.0036
9	sat	0.037942	0.961539	1.0000
12	sat	0.993368	0.006584	0.9981
17	sat	0.755344	0.231363	0.0000
19	sat	0.996412	0.003631	0.2270
22	sat	0.047274	0.952327	0.0000

8 Variable Model, (Alternate Sample)

Report	Actual Rating	(Neural Net Prediction) Y1	Y2	(Reg Pred) Y
14	exc	0.440171	0.565847	0.0000
1	exc	0.534030	0.477379	0.9999
11	exc	0.135683	0.865385	0.9993
15	exc	0.172818	0.829728	0.3845
6	exc	0.413235	0.588076	1.0000
2	exc	0.995555	0.004484	0.0323
5	sat	0.918370	0.081363	0.0000
9	sat	0.056906	0.941930	0.9998
12	sat	0.759408	0.232830	0.9910
17	sat	0.285728	0.721084	0.0000
19	sat	0.252400	0.744654	0.0000
22	sat	0.024315	0.975588	0.6596

13 Variable Model (Alternate Sample)

Report	Actual Rating	(Neural Net Prediction) Y1	Y2	(Reg Pred) Y
14	exc	0.389303	0.606516	1.0000
1	exc	0.597827	0.397503	1.0000
15	exc	0.092907	0.907757	0.9744
6	exc	0.780399	0.223840	0.0000
2	exc	0.944783	0.054456	0.9997
11	exc	0.066412	0.933905	0.9780
5	sat	0.954918	0.045763	0.0000
9	sat	0.022977	0.976975	1.0000
12	sat	0.018223	0.982116	0.0998
17	sat	0.902589	0.101709	0.0000
19	sat	0.114542	0.885549	0.0149
22	sat	0.024868	0.975444	0.9946

18 Variable Mode (Alternate Sample)

Report	Actual Rating	(Neural Net Prediction) Y1	Y2	(Reg Pred) Y
14	exc	0.731884	0.265040	0.4466
1	exc	0.303620	0.709906	0.0079
11	exc	0.037471	0.962645	0.9996
15	exc	0.092089	0.906209	0.9957
6	exc	0.933140	0.068237	0.7879
2	exc	0.990483	0.009197	0.00025
5	sat	0.737891	0.269006	0.0036
9	sat	0.037942	0.961539	1.0000
12	sat	0.993368	0.006584	0.9981
17	sat	0.755344	0.231363	0.0000
19	sat	0.996412	0.003631	0.2270
22	sat	0.047274	0.952327	0.0000

Appendix J: Rankit Plots

```
                    RANKITS VS (Net-Reg)
RANKITS
 3.0 +
     :
     :
     :                                     +
     :
 1.0 +                                     +
     :                         +           +
     :              +          +
     :              2
     :              2
-1.0 +              +
     :
     :   +
     :
     :
-3.0 +
     -+---------+---------+---------+---------+-
    -4.0      -1.0       2.0       5.0       8.0
                          (Net-Reg) X 10E-1
       APPROX. WILK-SHAPIRO 0.8490   12 CASES PLOTTED
```

Figure 12. Rankit Plot, Paired Differences; Orig Sample

```
                     RANKITS VS (Net-Reg)
  RANKITS
   3.0 +
       :
       :
       :                                      +
       :
   1.0 +                                      +
       :                           +          +
       :                           2
       :                +          +
       :                2
  -1.0 +    +
       :
       :    +
       :
       :
  -3.0 +
       -+---------+---------+---------+---------+-
      -8.0      -5.0      -2.0       1.0       4.0
                            (Net-Reg) X 10E-1
         APPROX. WILK-SHAPIRO 0.9125   12 CASES PLOTTED
```

Figure 13. Rankit Plot, Paired Differences; Alt Sample

Bibliography

1. Cameron, Kim. "A Study of Organizational Effectiveness and its Predictors," Management Science, 32: 87-112 (January 1986).

2. Cameron, Kim S. "Effectiveness As Paradox: Consensus and Conflict in Conceptions of Organizational Effectiveness," Management Science, 32: 539-553 (May 1986).

3. Dutta, Soumitra and Shashi Shekhar. "Bond Rating: A Non-Conservative Application of Neural Networks," Proceedings of the 1988 IEEE International Conference on Neural Networks, II-443 - II-450, IEEE Press, New York, 1988.

4. Fukushima, Kunihiko and others. "Neocognitron: a Neural Network Model for a Mechanism of Visual Pattern Recognition," IEEE Transactions on Systems, Man, and Cybernetics SMC-13: 826-834 (1983).

5. Hecht-Nielsen, Robert. "Theory of the Backpropagation Neural Network," An Invited Paper presented at the 1988 INNS Annual Meeting. HNC, Inc., San Diego CA, 1988.

6. Hecht-Nielsen, Robert. "Neurocomputing: Picking The Human Brain," IEEE Spectrum, 36-41 (March 1988).

7. Holt, James R. An Expert System Model of Organizational Climate and Performance. PhD dissertation. Texas A&M University, College Station TX, 1987.

8. Hopfield, J. J. and D. W. Tank. "Neural Computation of Decisions in Optimization Problems," Biological Cybernetics, 52: 141-152 (1985).

9. Hornik, Kurt and others. "Multi-layer Feedforward Networks are Universal Approximators," Paper Submitted to Neural Networks. UCSD Dept of Economics. (September 1988).

10. Jones, William P. and Josiah Hoskins. "Back-Propagation," Byte, 155-162 (October 1987).

11. Josin, Gary. "Neural-Network Heuristics," Byte, 183-192 (October 1987).

12. Lewin, Arie Y. and John W. Minton. "Determining Organizational Effectiveness: Another Look and an Agenda for Research," Management Science, 32: 514-537 (May 1986).

13. Lippmann, Richard P. "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, 36-54 (April 1987).

14. Minsky, Marvin L. and Seymour A. Papert. Perceptrons. Cambridge MA: MIT Press, 1969.

15. Montgomery, Douglas C. and Elizabeth A. Peck. Introduction to Linear Regression Analysis. New York: John Wiley & Sons, 1982.

16. Neter, John and others. Applied Linear Regression Models. Homewood Ill: Richard D. Irwin, inc, 1983.

17. Recce, Michael and Philip Treleavan. "Computing From The Brain," New Scientist, 61-64 (May 1988).

18. Rumelhart, David E. and James L. McClelland. Parallel Distributed Processing. Cambridge MA: MIT Press, 1986.

19. Weisberg, Sanford. Applied Linear Regression. New York: John Wiley & Sons, 1980.

Vita

Captain Franklin W. Baugh [redacted] He graduated from Georgia Military High School in Milledgeville, Georgia in 1979. Captain Baugh received a Bachelor of Science degree in Mechanical Engineering from Georgia Institute of Technology and was commissioned December 1983. He was assigned to the 321st Civil Engineering Squadron, Grand Forks AFB, North Dakota until January 1987. After serving a short tour with the U.S. Embassy in Muscat Oman, Captain Baugh entered the Graduate Engineering Management Program at the School of Systems and Logistics, Air Force Institute of Technology, in June 1988.

[redacted]